НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК 044.852

«До захисту допущено»
Завідувач кафедри СПСКС

_____ В.П.Тарасенко_
(підпис) (ініціали, прізвище)

"___" _____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
Системне програмування

на тему: Інтелектуальні засоби аналізу профілю користувача соціальної мережі

Виконав (-ла): студент (-ка) ІІ курсу, групи КВ-72мп
Анастасьєв Дмитро Вадимович

Науковий керівник доцент кафедри СПСКС, к.т.н., Замятін Д. С.

Рецензент _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики
Кафедра системного програмування і спеціалізованих комп'ютерних систем
Рівень вищої освіти – другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Системне програмування

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС
_____ _В.П.Тарасенко__
(підпис) (ініціали, прізвище)
«_ __»_____2018р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Анастасьєву Дмитру Вадимовичу

1. Тема дисертації: Інтелектуальні засоби аналізу профілю користувача соціальної мережі, науковий керівник дисертації Замятін Денис Станіславович доцент кафедри СПіСКС, к.т.н.

Затверджені наказом по університету від «30» жовтня 2018 р. №4030-с

2. Термін подання студентом дисертації 07 грудня 2018 р.

3. Об'єкт дослідження класифікація користувачів соціальної мережі Instagram за гендерною ознакою

4. Предмет дослідження методи аналізу та класифікації тексту та зображень

5. Перелік завдань, які потрібно розробити
   - опис предметної області досліджень та обґрунтування методів класифікації тексту та зображень;
   - знаходження важливих ознак для визначення статі користувачів;
   - вибір на налаштування ефективної моделі нейронної мережі.

6. Перелік ілюстративного матеріалу
   - Схеми нейронних мереж
   - Таблиця результатів
   - Використані формули
   - Презентація
7. Перелік публікацій

– «Класифікація імен за гендерною ознакою», XI конференція молодих вчених ПМК-2018-2. – 2018;
– «Методи аналізу статі користувачів соціальних мереж», XIV міжнародна науково-практична конференція “Наука та освіта без меж”– 2018;

8. Дата видачі завдання 5 вересня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Вивчення літератури за тематикою проекту	05.09.2017	
2	Аналіз існуючих рішень	20.01.2018	
3	Підготовка матеріалів першого розділу магістерської дисертації	09.03.2018	
4	Підготовка матеріалів другого розділу магістерської дисертації	30.04.2018	
5	Підготовка матеріалів третього розділу магістерської дисертації	10.09.2018	
6	Підготовка графічної частини дипломного проекту	16.10.2018	
7	Оформлення документації дипломного проекту	01.11.2018	
8	Попередній розгляд магістерської дисертації на кафедрі	26.11.2018	

Студент _____ Анастасьєв Д.В.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Замятін Д.С.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Актуальність теми. Різноманітні соціальні мережі з кожним днем збільшують свою популярність, стаючи одними з найвідвідуваніших інтернет ресурсів. Створюючи профілі на таких ресурсах, користувачі надають у відкритий доступ багато даних про себе: вік, стать, ім’я, уподобання і т.д. Аналізуючи ці дані можливо класифікувати користувачів соціальних мереж за різними критеріями, а також передбачати наявність певних характеристик користувачів. Такий аналіз даних може бути широко використаний у багатьох економічних, технічних на наукових сферах, наприклад у сфері маркетингу для впровадження таргетованої реклами або у сфері рекрутменту для знаходження кваліфікованих фахівців.

Об’єктом дослідження є класифікація користувачів соціальної мережі Instagram за гендерною ознакою

Предметом дослідження є методи аналізу та класифікації тексту та зображень

Методи дослідження – методи машинного навчання для вирішення задач класифікації текстових даних та зображень

Мета роботи: досягнення якомога вищого відсотка правильного передбачення статі користувачів соціальної мережі Instagram, використовуючи різноманітні дані профілів цих користувачів

Наукова новизна одержаних результатів полягає в наступному:

1. Проаналізовано різні методи машинного навчання для вирішення задач класифікації зображень та текстових даних; вибрано ключові характеристики користувачів соціальної мережі Instagram, за якими можна визначити стать; створено базу цільових даних для навчання

нейронних згорткових мереж; проаналізовано декілька моделей нейронних мереж, обрано та налаштовано найбільш ефективну.

2. Створено інтелектуальну систему класифікації користувачів соціальної мережі Instagram за гендерною особливістю, використовуючи дані з профілів та методи аналізу описані вище.

Практична цінність одержаних в роботі результатів полягає в тому, що запропонований спосіб дозволяє ефективно, класифікувати користувачів соціальних мереж за ознаками які не вказані у профілі користувача, тобто передбачувати наявність певних ознак, використовуючи наявні ознаки.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на:

- XI науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018-2 (Київ, 15-17 листопада 2018 р.);

- XIV міжнародній науково-практична конференції "Наука та освіта без меж"– 2018;

Публікації. За результатами дослідження опубліковано 2 наукові праці, з них 1 стаття та 1 тези конференції.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів та висновків.

У вступі подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи

У першому розділі розглянуто теоретичні відомості по заданій темі, а також проведений аналіз, який дає змогу визначити основні

переваги та недоліки існуючих способів класифікації зображень та текстових даних.

У другому розділі наведено опис проведених модифікацій до існуючих методів аналізу даних та описано способи їх використання

У третьому розділі наведено опис проведених вимірювань точності роботи запропонованих методів та опис створеної системи класифікації

У висновках представлені результати проведеної роботи.

Робота представлена на **90** аркушах, містить **17** рисунків, **1** таблицю та список використаних літературних джерел з **9** найменувань.

Ключові слова: Keras, нейронні мережі, згорткові нейронні мережі, машинне навчання

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

БД	База даних
РБД	Реляційна база даних
API	Прикладний програмний інтерфейс
Python	Мова програмування високого рівня
ПЗ	Програмне забезпечення
PyCharm	Інтегроване середовище розробки для мови програмування Python
SQL	Structural query language – структурована мова запитів до реляційних баз даних
ОС	Операційна система
ГС	Градієнтний спуск
СГС	Стохастичний градієнтний спуск
ЗПМ	Зворотне поширення помилки
Softmax	Функція активації
MNIST	База даних зразків рукописних цифр

ВСТУП

На даний момент одними з найпопулярніших інтернет ресурсів є соціальні мережі. Кожен день мільйони користувачів обмінюються різноманітною інформацією використовуючи улюблені сайти, такі як Instagram.com, Facebook.com, VK.com та інші. Сервери цих соціальних мереж налічують терабайти інформації, яка знаходиться у відкритому доступі. Аналізуючи інформацію про користувачів соціальних мереж їх можливо класифікувати та відносити до певних груп за будь-якими характеристиками: вік, стать, місце проживання, професійні особливості, уподобання та інші. Використовуючи сучасні технології машинного навчання можливо навіть передбачати наявність певних характеристик у користувачів використовуючи інформацію з профілів.

Такий аналіз даних може бути широко використаний у багатьох економічних, технічних на наукових сферах, наприклад у сфері маркетингу для впровадження таргетованої реклами або у сфері рекрутменту для знаходження кваліфікованих фахівців.

У даній роботі розглядається розробка системи класифікації користувачів мережі Instagram за гендерною особливістю: дослідження вхідних параметрів для обробки, методів аналізу текстових даних та зображень. Метою розробки є досягнення якомога вищого відсотка правильного передбачення статі користувачів соціальної мережі Instagram, використовуючи різноманітні дані профілів цих користувачів.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОГЛЯД МЕТОДІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ТА ТЕКСТОВИХ ДАНИХ

Для вирішення проблеми класифікації профілів соціальних мереж за гендерною ознакою вирішено використовувати алгоритми машинного навчання. Існує багато методів машинного навчання для аналізу текстових даних та зображень, серед яких потрібно обрати ті, які найкраще підходять для аналізу даних які розглядаються у роботі. Серед усіх ознак профілів соціальної мережі Instagram обрано наступні: нік користувача, повне ім’я, фотографія профілю та декілька останніх фотографій на сторінці.

1.1. ПОНЯТТЯ ПЕРСЕПТРОНА

Нейронні мережі та глибоке навчання зараз забезпечують найкращі рішення для багатьох проблем розпізнавання образів, розпізнавання мови та обробки природної мови. Зорова система людини є складною та досконалою системою нашого організму. У кожній півкулі нашого мозку люди мають первинні зони зорової кори, також відомі як V1, що містить 140 мільйонів нейронів, з десятками мільярдів зв'язків між ними. І все-таки людське бачення передбачає не тільки V1, але і цілий ряд візуальних шарів: V2, V3, V4 і V5, здійснює поступово більш складну обробку зображень. В нашій голові знаходиться суперкомп'ютер, налаштований еволюцією протягом сотень мільйонів років і прекрасно пристосований для розуміння візуального світу. Люди добре розуміють, що показують їх очі. Але майже вся ця робота виконується несвідомо. І тому ми не оцінюємо наскільки складна проблема, яку вирішують наші візуальні системи. Складність розпізнавання зображень стає очевидною, якщо ми спробуємо написати комп'ютерну програму для розпізнавання цифр. Те, що здається легким, раптово стає надзвичайно складними. Проста інтуїція про те, як ми визнаємо фігури - "у 9 є петля зверху та вертикальна дуга в правому нижньому куті" - виявляється не так просто алгоритмічно. Нейронні мережі підходять до проблеми по-іншому. Ідея полягає в тому, щоб взяти велику

кількість рукописних цифр, відомих як приклади навчання, а потім розробити систему, яка може навчатися на прикладах навчання. Іншими словами, нейронна мережа використовує приклади для автоматичного визначення правил розпізнавання рукописних цифр. Крім того, збільшуючи кількість навчальних прикладів, мережа зможе дізнатись більше про почерк, що підвищить її точність. Таким чином, якщо показати лише 100 навчальних цифр, можливо створити гірший класифікатор почерку, ніж використовуючи тисячі або навіть мільйони навчальних прикладів.

Що таке нейронна мережа? Для початку потрібно описати тип штучного нейрона, який називається персептрон. Персептони були розроблені в 1950-х і 1960-х роках вченим Френком Розенблаттом, натхненні ранніми роботами Уоррена МакКуллоха та Вальтера Пітта. Сьогодні частіше вживаються інші моделі штучних нейронів, в сучасній роботі над нейронними мережами використовується головна нейронна модель, яку називають сигмовидним нейроном. Щоб зрозуміти, чому сигмоподібні нейрони визначаються так, як вони є, варто проаналізувати перцептрони.

Отже, як працюють персептрони? Персептрон приймає кілька подвійних входів, x1, x2, ..., і видає єдиний бінарний вивід (рис. 1).

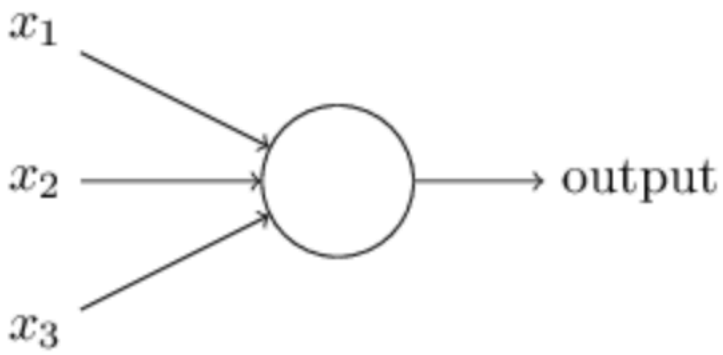

Рисунок 1 - Простий персептрон

У показаному прикладі персептрон має три входи: x1, x2, x3. Взагалі, це може мати більше або менше входів. Розенблат запропонував просте правило для обчислення результату. Він ввів ваги, w1, w2, ..., реальні

цифри, що виражають важливість відповідних входів на виході. Вихід нейрона, 0 або 1, визначається, чи є вагова сума $\sum_j wj\,xj$ меншою або більшою за деяке порогове значення. Подібно вагам, порогове значення - це дійсне число, яке є параметром нейрона. Це основна математична модель. Спосіб, яким ми можемо думати про персептрон, полягає в тому, що це пристрій, який приймає рішення, зважуючи свідчення. Наведемо простий але зрозумілий приклад. Припустимо, у вашому місті відбудеться фестиваль сирів. Ви любите сир, і намагаєтеся вирішити, чи їхати на фестиваль чи ні. Ви можете прийняти рішення, зваживши три фактори: наскільки хороша погода, чи буде хтось вас супроводжувати, чи знаходиться фестиваль біля громадського транспорту. Ми можемо представити ці три фактори за допомогою відповідних двійкових змінних x1, x2 та x3. Наприклад, ми маємо x1 = 1, якщо погода хороша, а x1 = 0, якщо погода погана. Аналогічно, x2 = 1, якщо ваш хлопець або подруга хоче піти, а x2 = 0, якщо ні. І аналогічно знову для x3 і громадського транспорту. Тепер припустимо, що ви дуже любите сир, можете піти на фестиваль, навіть якщо ваш хлопець чи дівчина не зацікавлена і важко доїхати до фестивалю. Але, можливо, ви дійсно засмучуєтеся через погану погоду, і немає жодного способу, що ви підете на фестиваль, якщо погана погода. Ви можете використовувати персептрони для моделювання такого роду процесу прийняття рішень. Один із способів зробити це - вибрати вагу w1 = 6 для погоди, а w2 = 2 і w3 = 2 для інших умов. Чим більше значення w1 вказує на те, що для вас погода важливіша, ніж чи приєднається ваш хлопець чи подруга до вас або близькість громадського транспорту. Нарешті, припустимо, що ви вибрали пороговий показник 5 для персептрона. За допомогою цього вибору персептрон реалізує бажану модель прийняття рішень, виводячи 1, коли погода хороша, і 0, коли погана погода. Не має значення, чи ваш хлопець чи подруга зможе піти з вами та близькість громадського транспорту. Змінюючи ваги та порогові значення, ми можемо отримати різні моделі прийняття рішень. Наприклад, припустімо, що ми замість цього вибрали порогове значення 3. Тоді

персептрон вирішив, що ви повинні піти на фестиваль кожного разу, коли погода була хороша, або коли обидва фестивалі були біля громадського транспорту, а ваш хлопець чи дівчина хотіли приєднатися до вас. Іншими словами, це буде інша модель прийняття рішень. Зниження порогу означає, що ви більше бажаєте піти на фестиваль. Очевидно, персептрон не є повною моделлю прийняття рішень людьми. Але приклад ілюструє, як персептрон може зважити різні види доказів для прийняття рішень. І це повинно здатися правдоподібним, що складна мережа персептронів могла б зробити досить витончені рішення (рис. 2).

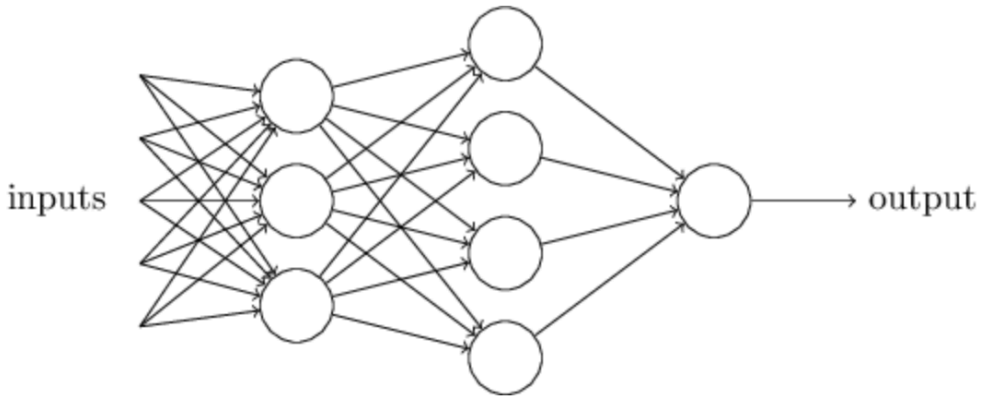

Рисунок 2 - Складна мережа персептронів

У цій мережі перша колонка персептронів - що ми називатимемо першим шаром персептронів - приймає три дуже прості рішення, зважуючи вхідні докази. А як щодо персептронів у другому шарі? Кожен з цих персептронів приймає рішення, зважуючи результати першого рівня прийняття рішень. Таким чином, персептрон у другому шарі може приймати рішення на більш складному та більш абстрактному рівні, ніж персептрони в першому шарі. І навіть більш складні рішення можуть бути зроблені персептроном у третьому шарі. Таким чином, багаторівнева мережа персептронів може приймати складні рішення. Раніше визначалося, що персептрон має лише один вихід. Мережі з персептронами виглядають так, що вони мають кілька виходів. Фактично, вони все одно мають єдиний вихід. Багатократні вивідні стрілки - це просто корисний спосіб показати, що вихідний сигнал з

персептрона використовується як вхід для декількох інших персептронів. Умова $\sum_j wj\ xj >$ поріг є громіздким, і ми можемо зробити дві зміни, щоб спростити його. Перша зміна полягає в тому, щоб написати $\sum_j wj\ xj$ як, $w * x = \sum_j wj\ xj$, де w та x - це вектори, компоненти яких є вагами та входами, відповідно. Друга зміна полягає в тому, щоб перемістити поріг на іншу сторону нерівності і замінити його тим, що називається упередженням персептрона, b = - порогом. Використовуючи зміщення замість порогу, правило персептрон можна переписати:

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

Можна подумати про упередженість як міру того, як легко отримати персептрон для виведення 1. Для персептрона з дуже великим ухилом, перцептрон надзвичайно простий для виведення 1. Але якщо зміщення дуже негативне, для персептрона важко вивести 1. Очевидно, що введення упередженості є лише невеликою зміною в те як ми описуємо персептрони, але пізніше ми побачимо, що це призводить до подальших умовних спрощень. Описано персептони як метод зважування доказів для прийняття рішень. Інший спосіб перцептронів може бути використаний для обчислення елементарних логічних функцій, які ми зазвичай вважаємо основними для обчислення, такі функції, як AND, OR та NAND. Наприклад, припустимо, у нас є персептрон з двома входами, кожен з вагою -2, а загальний ухил у 3 (рис. 3).

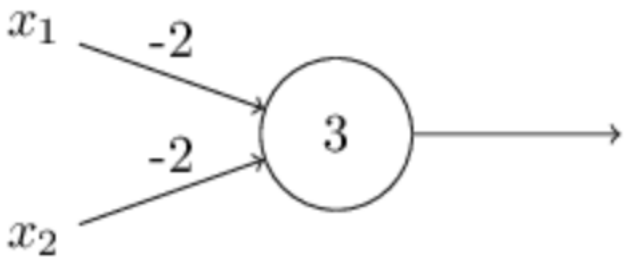

Рисунок 3 - Персептрон з двома входами, кожен з вагою -2 та з порогом 3

Тоді ми бачимо, що вхід 00 породжує вихід 1, оскільки (-2) * 0 + (- 2) * 0 + 3 = 3 є позитивним. Вводимо символ *, щоб зробити явне множення. Подібні розрахунки показують, що входи 01 і 10 породжують вихід 1. Але вхід 11 породжує вихід 0, оскільки (-2) * 1 + (- 2) * 1 + 3 = -1 є негативним. Отже, наш персептрон реалізує функцію NAND. Приклад NAND показує, що ми можемо використовувати персептрони для обчислення простих логічних функцій. Фактично, ми можемо використовувати мережі персептронів для обчислення будь-якої логічної функції взагалі. Причиною є те, що функція NAND є універсальною для обчислень, тобто ми можемо побудувати будь-яке обчислення з NAND функцією. Наприклад, ми можемо використовувати функцію NAND для побудови схеми, яка додає два біти, x1 та x2. Для цього потрібно обчислювати побітну суму x1 $\oplus$ x2, а також біт переносу, який дорівнює 1, коли обидва x1 та x2 дорівнюють 1, тобто біт переміщення - це лише побітовий продукт x1x2 (рис. 4).

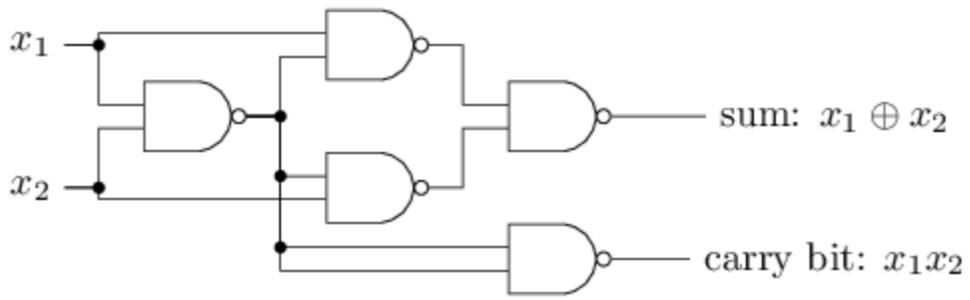

Рисунок 4 - Реалізація логічних операцій

Щоб отримати еквівалентну мережу персептронів, ми замінюємо всі функції NAND на персептрони з двома входами, кожний з вагою -2, а загальний зміщення в 3.

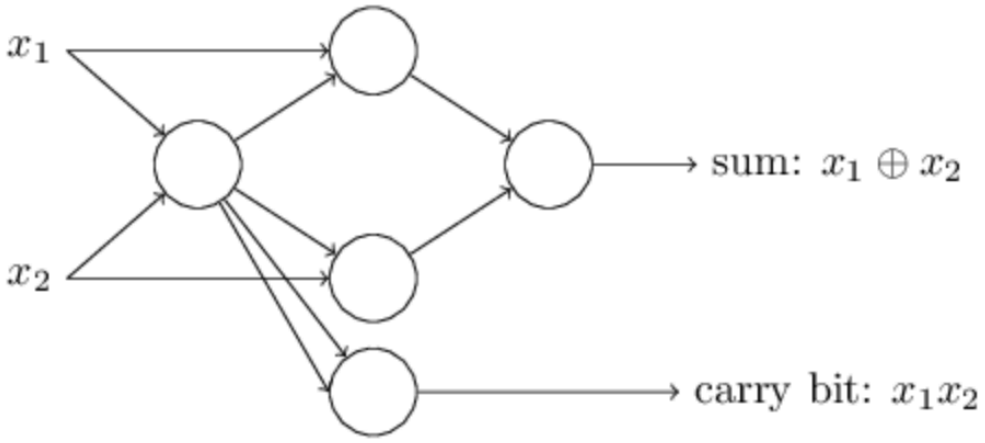

Рисунок 5 - Результуюча мережа

Одним з помітних аспектів цієї мережі персептронів є те, що вихід з лівого персептрону використовується вдвічі як вхідний в нижній частині персептрона. Насправді це не має значення. Якщо ми не хочемо дозволити подібні речі, то можна просто об'єднати дві лінії в єдине з'єднання з вагою -4 замість двох зв'язків з -2 вагами. З цією зміною мережа виглядає наступним чином, при всій немаркованої вазі, рівній -2, всі упередження рівні 3, а одинична вага -4, як зазначено. До цього часу входи, такі як x1 та x2, визначалися як змінні, що плавають ліворуч від мережі персептронів. Фактично, для кодування входів використовується додатковий шар персептронів - вхідний шар. Ця позначка для вхідних персептронів, в якій ми маємо вихід, але не має вхідних даних, є скороченням. Це насправді не означає персептрон без вхідних даних. Щоб побачити це, припустимо, у нас був персептрон без вхідних даних. Тоді вагова сума $\sum_j wj\,xj$ завжди дорівнює нулю, і тому персептрон виведе 1, якщо b> 0, а 0, якщо b≤0. Тобто, персептрон просто виведе фіксоване значення, а не бажане значення (x1 у наведеному вище прикладі). Краще думати про вхідні персептрони, оскільки вони взагалі не перцептрони, а лише спеціальні одиниці, які просто визначаються для виведення бажаних значень, x1, x2 і так далі. Приклад суматора показує, як мережа персептронів може бути використана для імітації схеми, що містить багато функцій NAND. І тому, що функції

NAND є універсальними для обчислення, випливає, що персептони також універсальні для обчислень.

Обчислювальна універсальність перцептронів одночасно заспокоює і розчаровує. Оскільки це означає, що мережі персептронів можуть бути настільки ж потужними, як і будь-які інші обчислювальні пристрої. Але це також викликає розчарування, оскільки це змушує здаватися, ніби персептрони є просто новим типом функції NAND. Проте ситуація є кращою, ніж передбачає така думка. Виявляється, ми можемо розробити алгоритми навчання, які можуть автоматично налаштовувати ваги та упередження мережі штучних нейронів. Це налаштування відбувається у відповідь на зовнішні подразники, без безпосереднього втручання програміста. Ці алгоритми навчання дозволяють нам використовувати штучні нейрони таким чином, що радикально відрізняється від звичайних логічних функцій. Замість того, щоб явно викласти схему NAND та інших функцій, наші нейронні мережі можуть просто навчитися вирішувати проблеми, а іноді і проблеми, для яких було б надзвичайно важко безпосередньо проектувати звичайну схему.

1.2. СИГМОВИДНІ НЕЙРОНИ

Припустимо, у нас є мережа персептронів, які ми хотіли б використати, щоб навчитися вирішувати певну проблему. Наприклад, входи до мережі можуть бути необоротними піксельними даними зі сканованого рукописного зображення цифри. І ми хочемо, щоб мережа вивчала вага та упередження, щоб вихід з мережі правильно класифікував цифру. Щоб побачити, як навчання може працювати, припустімо, ми внесемо невеликі зміни в певну вагу (або зміщення) в мережі. Те, що ми хотіли б, для цієї невеликої зміни ваги, викликати невелику відповідну зміну на виході з мережі (рис. 6).

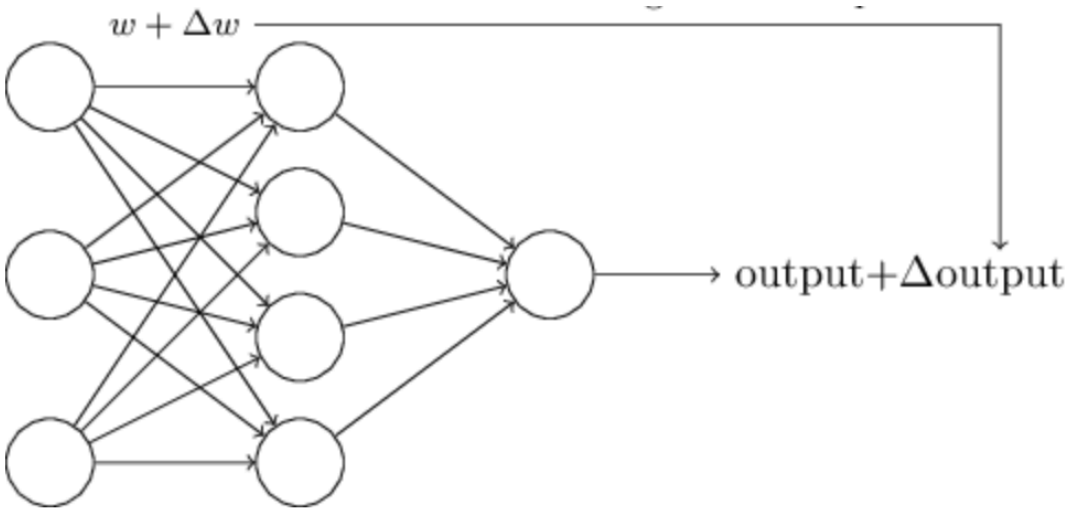

Рисунок 6 - Найпростіший алгоритм навчання

Невелика зміна ваги (або зміщення) викликає лише невелику зміну результату, ми можемо використати цей факт, щоб змінити ваги та зміщення, щоб наша мережа поводилася так, як ми хочемо. Наприклад, припустимо, що мережа помилково класифікувала зображення як "8", коли вона повинна бути "9". Ми могли б зрозуміти, як зробити невеликі зміни в вагах і упередженнях, щоб мережа дещо наблизилась до класифікації зображення як "9". І тоді ми повторимо це, змінюючи ваги та тенденції знову і знову, щоб виробляти краще і краще продуктивність. Мережа буде вчитися. Проблема в тому, що це не те, що відбувається, коли наша мережа містить персептрони. Фактично, невелика зміна ваг або зміщення будь-якого окремого персептрона в мережі іноді може призвести до того, що вихід цього персептрона повністю перевертається, скажімо, від 0 до 1. Це може призвести до поведінки решти мережі повністю змінюється в якомусь дуже складному вигляді. Таким чином, якщо ваша "9" тепер може бути класифікована правильно, поведінка мережі на всіх інших зображеннях, ймовірно, повністю змінилася в деякому важко контролювати спосіб. Це ускладнює спостереження за тим, як поступово змінювати ваги та упередження, щоб мережа наближалася до бажаної поведінки. Можливо, є якийсь розумний спосіб подолати цю проблему. Але це не відразу очевидно, як ми можемо отримати мережу персептронів для навчання. Ми можемо

подолати цю проблему, ввівши новий тип штучного нейрона, який називається сигмовидним нейроном. Сигмоїдні нейрони подібні до персептронів, але модифіковані таким чином, що невеликі зміни їх ваг і зміщення викликають лише невелику зміну їх виходу. Це найважливіший факт, який дозволить вивчити мережу сигмовидних нейронів. Подібно до персептрону, сигмовидний нейрон має вхідні дані, x1, x2, ... Але замість того, щоб бути лише 0 або 1, ці входи також можуть приймати будь-які значення між 0 і 1. Так, наприклад, 0.638 ... є дійсним входом для сигмовидного нейрона. Також, як і перцептрон, сигмоподібний нейрон має ваги для кожного вхідного сигналу, w1, w2, ..., а також загального упередження b. Але висновок не 0 або 1. Замість цього це $\sigma (w \cdot x + b)$, де σ називається сигмоїдною функцією. З першого погляду сигмоподібні нейрони виявляються сильно відмінні від персептронів. Насправді, між персептронами і сигмовидними нейронами багато подібностей, а алгебраїчна форма сигмоїдної функції виявляється більш технічною деталлю, ніж справжній бар'єр для розуміння. Щоб зрозуміти схожість з персептронной моделлю, припустимо, $z \equiv w \cdot x + b$ - велике позитивне число. Тоді $e – z \approx 0$ і так $\sigma (z) \approx 1$. Іншими словами, коли $z = w \cdot x + b$ є великим і позитивним, вихід із сигмоподібного нейрона приблизно 1, як це було б для персептрона. Припустимо, з іншого боку, що $z = w \cdot x + b$ дуже негативно. Тоді $e - z \rightarrow \infty$, а $\sigma (z) \approx 0$. Отже, коли $z = w \cdot x + b$ дуже негативно, поведінка сигмоподібного нейрона також тісно апроксимує персептрон. Тільки тоді, коли $w \cdot x + b$ має скромний розмір, існує значне відхилення від моделі персептрона. Насправді, точна форма σ не настільки важлива - що дійсно важливо, це форма функції при побудові (Рис. 7).

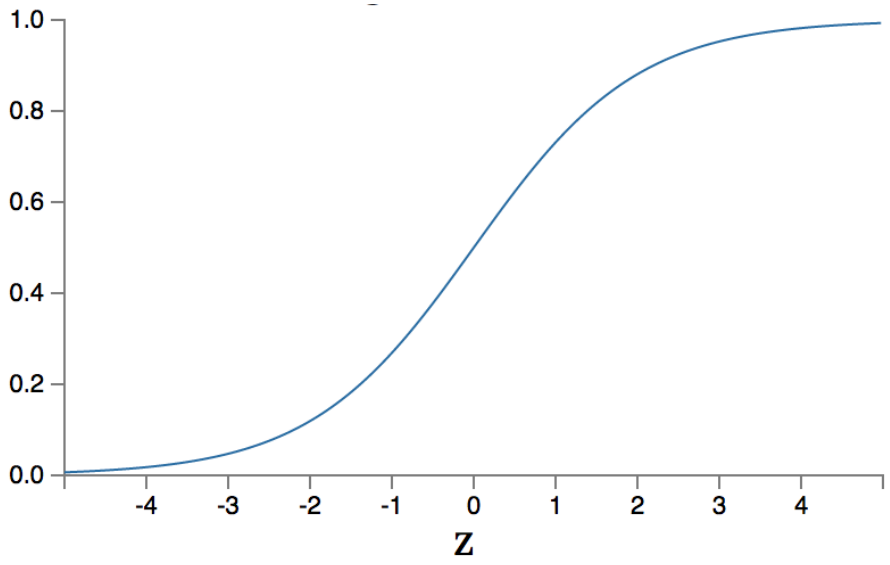

Рисунок 7 - Графік сигмоїдної функції

1.3. МЕТОДИ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

Для класифікації зображень використовують мережі, які розбивають дуже складне питання у дуже прості питання, відповідальні на рівні окремих пікселів. Це робиться за допомогою серії багатьох шарів, причому ранні верстви відповідають дуже простим і конкретним питанням про вхідний образ, а пізніше - шари, що створюють ієрархію все більш складних та абстрактних концепцій. Мережі з такою багаторівневою структурою - два або більше прихованих шарів - називаються глибинними нейронними мережами.

З 2006 року розроблено комплекс методик, що дозволяють вивчати глибокі нейронні мережі. Ці глибокі методи навчання засновані на стохастичному градієнтному спуску та зворотному поширенні, але також вводять нові ідеї. Ці методи дали змогу набагато більш глибоким (і більшим) мережам навчатись. Зараз люди регулярно тренують мережі з 5-10 прихованими шарами. І, виявляється, вони працюють набагато краще, ніж на дрібних нейронних мережах, тобто мережах з лише одним прихованим шаром. Причиною, звичайно, є здатність глибоких мереж створити складну ієрархію понять. Це трохи подібно до того, як звичайні

мови програмування використовують модульний дизайн та ідеї щодо абстракції, що дозволяє створювати складні комп'ютерні програми. Порівняння глибокої мережі з мілководною мережею трохи нагадує порівняння мови програмування з можливістю виклику функцій на стиснуту мову без можливості робити такі виклики. Абстракція займає іншу форму в нейронних мережах, ніж у звичайних програмуваннях, але це так само важливо.

В останні кілька років нейронні мережі використовуються в усіх галузях машинного навчання, але найбільший фурор вони безперечно зробили в області комп'ютерного зору. В рамках змагань ImageNet було представлено безліч різних архітектур згорткових мереж. Щоб поліпшити якість розпізнавання своїх мереж, дослідники намагалися додавати в мережі більше шарів, проте з часом прийшло розуміння, що іноді обмеження продуктивності просто не дозволяють навчати і використовувати настільки глибокі мережі. Це стало мотивацією для використання depthwise separable convolutions і створення архітектури Xception. У 2015 році була запропонована архітектура Inception, ідея якої полягала в наступному: замість того, щоб вибирати розмір ядра, візьмемо кілька варіантів відразу, використовуємо їх всі одночасно і об’єднаємо результати. Однак це суттєво збільшує кількість операцій, які необхідно виконати для обчислення активацій одного шару, тому пропонується таку хитрість: перед кожним згортковим блоком робити згортку з розміром ядра 1x1, знижуючи розмірність сигналу, що подається на вхід згортками з великими розмірами ядер. Звичайний згортковий шар одночасно обробляє як просторову інформацію (кореляцію сусідніх точок всередині одного каналу), так і міжканальну інформацію, так як згортка застосовується до всіх каналів відразу. Архітектура Xception базується на припущенні про те, що ці два види інформації можна обробляти послідовно без втрати якості роботи мережі, і розкладає звичайну згортку на pointwise convolution (яка обробляє тільки міжканальну кореляцію) і spatial convolution (яка обробляє тільки просторову кореляцію в рамках окремого каналу) .

Подивимося на реальний ефект. Для порівняння візьмемо дві по-справжньому глибоких архітектури згорткових мережі - ResNet50 і InceptionResNetV2. ResNet50 має 25 636 712 ваг, а предобученная модель в Keras важить 99 Мб. Точність, яка досягається цією моделлю на датасета ImageNet, становить 75,9%. InceptionResNetV2 має 55 873 736 учнів параметрів і важить 215 Мб, досягаючи точності 80.4%. Що ж виходить з архітектурою Xception? Мережа має 22 910 480 ваг і важить 88 Мб. При цьому точність класифікації на ImageNet становить 79%.

Таким чином, ми отримуємо архітектуру мережі, яка перевершує за точністю ResNet50 і лише трохи поступається InceptionResNetV2, при цьому істотно виграючи за розмірами, а значить по необхідних ресурсів як для навчання, так і для використання цієї моделі.

1.4. ГЛИБОКЕ НАВЧАННЯ

Глибокі нейронні мережі часто набагато важче тренувати, ніж дрібні нейронні мережі. Це не зовсім так, оскільки у нас є вагомі підстави вважати, що, якщо б ми могли б тренувати глибокі мережі, вони були б набагато потужніші, ніж дрібні сітки. Розглянемо методи, які можуть бути використані для навчання глибинних мереж, і застосовувані на практиці. Ми також розглянемо більш широку картину, коротко розглядаючи останні досягнення у використанні глибоких мереж для розпізнавання образів, розпізнавання мовлення та інших програм. І ми проведемо короткий, погляд на те, що може мати майбутнє для нейронних мереж, а також для штучного інтелекту.

Раніше ми використовували мережі, в яких суміжні шари мережі повністю підключені один до одного. Тобто, кожен нейрон у мережі підключений до кожного нейрона в суміжних шарах. Зокрема, для кожного пікселя у вхідному зображенні ми кодували інтенсивність пікселя як значення для відповідного нейрона у вхідному шарі. Наприклад для зображень 28 × 28 пікселів, які ми використовували, це означає, що наша

мережа має 784 (28 × 28) вхідних нейронів. Але після рефлексії, дивно, що для класифікації зображень використовують мережі з повністю підключеними шарами. Причина в тому, що така архітектура мережі не враховує просторову структуру зображень. Наприклад, він обробляє вхідні пікселі, які знаходяться далеко один від одного і близькі один до одного на однаковій основі. Подібні поняття просторової структури повинні, натомість, бути виведені з навчальних даних. Але що робити, якщо замість того, щоб починати з мережевої архітектури, яка є таблицею, ми використовували архітектуру, яка намагається скористатися перевагами просторової структури? Ці мережі використовують спеціальну архітектуру, яка особливо добре адаптована для класифікації зображень. Використання цієї архітектури дозволяє швидко тренувати згорткові мережі. Це, в свою чергу, допомагає нам готувати глибокі, багатошарові мережі, які дуже добре класифікують зображення. Сьогодні в більшості нейронних мереж для розпізнавання зображень використовуються глибокі згорткові мережі або близький варіант. Згорткові нейронні мережі використовують три основні ідеї: місцеві рецептивні поля, загальні ваги та об'єднання. Давайте розглянемо кожну з цих ідей у свою чергу.

У повністю з'єднаних шарах, показаних раніше, входи були зображені як вертикальна лінія нейронів. У згортковій мережі зручніше замість вектору застосовувати матрицю, як квадрат з 28 × 28 нейронів, значення яких відповідають 28 × 28 пікселів інтенсивності, які ми використовуємо як входи (рис. 8)

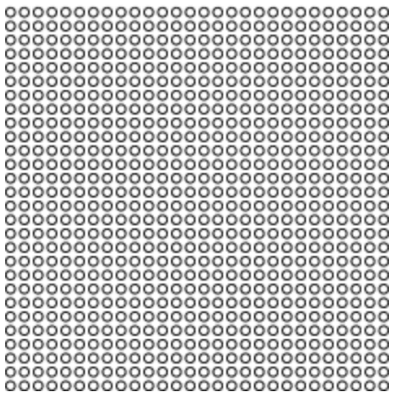

Рисунок 8 - Вхідний шар нейронів згорткової мережі

Як звичайно, ми підключимо вхідні пікселі до шару прихованих нейронів. Але ми не будемо з'єднувати кожний вхідний піксель з кожним прихованим нейроном. Замість цього ми створюємо лише з'єднання в невеликих, локалізованих регіонах вхідного зображення. Точніше, кожен нейрон у першому прихованому шарі буде з'єднаний з невеликою ділянкою вхідних нейронів, скажімо, наприклад, область 5 × 5, яка відповідає 25 вхідних пікселів. Отже, для певного прихованого нейрона, ми можемо мати зв'язки, які виглядають так, як забражено на рис. 9.

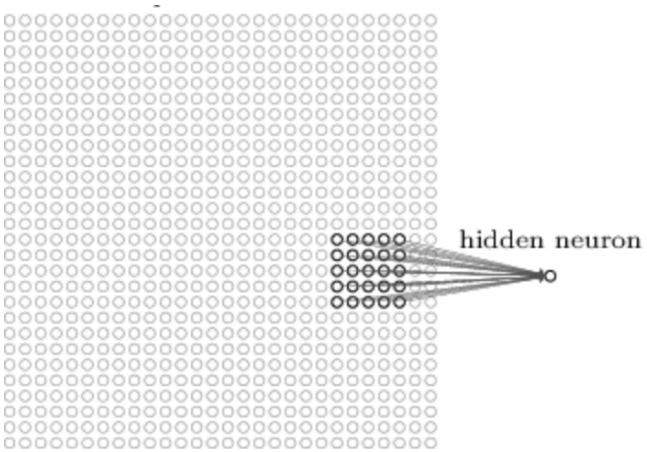

Рисунок 9 - Згортковий шар мережі

Цей регіон у вхідному зображенні називається локальним сприйнятливим полем для прихованого нейрона. Це маленьке вікно на вхідних пікселях. Кожне з'єднання вивчає вагу. І прихований нейрон також вивчає загальне упередження. Ви можете подумати про цей особливий прихований нейрон, як навчаючись аналізувати його специфічне місцеве сприйняття.

Потім ми просушуємо локальне сприйняткове поле по всьому вхідному зображенню. Для кожного локального рецептивного поля існує інший прихований нейрон у першому схованому шарі. Якщо у нас є 28 × 28 вхідних зображень і 5 × 5 локальних сприйнятливих полів, у прихованому шарі буде 24 × 24 нейронів. Це пов'язано з тим, що ми можемо лише

переміщати місцеве сприйнятне поле 23 нейронами вперед (або 23 нейронами вниз), перш ніж зіткнутися з правою (або нижньою) частиною вхідного зображення. Таким чином показано, що локальне сприйняткове поле переміщується на один піксель одночасно. Фактично, іноді використовується інша довжина кроку. Наприклад, ми можемо перемістити локальне сприйняткове поле на 2 пікселі вправо (або вниз), в цьому випадку ми скажемо, що використовується довжина кроку 2. У цьому розділі ми в основному будемо дотримуватися довжини кроку 1, однак варто знати, що люди іноді експериментують із різною довжиною кроку. З цієї причини ми іноді називаємо карту з вхідного шару до прихованого шару картою властивостей. Ми називаємо ваги, що визначають карту об'єктів, спільними вагами. І ми називаємо упередження, що визначає карту властивостей таким чином, спільним упередженням. Частина ваг і зміщення часто визначають ядро або фільтр. Структура мережі, яку описана до теперішнього часу, може виявити лише один вид локалізованої функції. Для розпізнавання зображень нам знадобиться більше, ніж одна функціональна карта. Таким чином, повний згортковий шар складається з декількох карток властивостей. У наведеному прикладі є 3 функціональні карти. Кожна карта функцій визначається сукупністю 5 × 5 загальних ваг і єдиним загальним ухилом. Результат полягає в тому, що мережа може виявляти 3 різних видів функцій, при цьому кожна функція може бути виявлена по всьому зображенню.

Я показав лише 3 функціональні карти, щоб зберегти схему вище. Проте, на практиці згорткові мережі можуть використовувати більше (і, можливо, багато інших) функціональних карт. Одна з ранніх згорткових мереж, LeNet-5, використовувала 6 функціональних карт, кожна з яких зв'язана з 5 × 5 локальним рецептивним полем, для розпізнавання цифр MNIST. Таким чином, приклад, наведений вище, насправді дуже близький до LeNet-5.

1.5. МЕТОДИ КЛАСИФІКАЦІЇ ТЕКСТОВИХ ДАНИХ

У роботі розглядається проблеми бінарної класифікації текстових даних а саме імен та ніків користувачів соціальних мереж. Для цього можливо просто використовувати словники. Перевагами такого методу є точна класифікація імен, які збережені в словниках, але великим недоліком такого підходу є абсолютна неспроможність класифікації імен, які не входять до словників. Таким чином доцільно використати алгоритми машинного навчання для класифікації імен.

Одним з найважливіших факторів від якого залежить успішність використання обраного методу є правильний підбір вхідних ознак. В даному випадку потрібно знайти такі ознаки імен, за якими можна дійсно класифікувати їх як жіночі або чоловічі. Для імен це зробити досить складно, адже гендерні ознаки імен відрізняються для різних культур та національностей. Наприклад, слов’янські жіночі імена частіше закінчуються на голосну букву ніж чоловічі, чого точно не можна сказати про імена інших національностей.

Також для вирішення цієї задачі можливо використовувати обробку природної мови - загальний напрям інформатики, штучного інтелекту та математичної лінгвістики. Він вивчає проблеми комп'ютерного аналізу та синтезу природної мови. Стосовно штучного інтелекту аналіз означає розуміння мови, а синтез — генерацію розумного тексту. Розв'язок цих проблем буде означати створення зручнішої форми взаємодії комп'ютера та людини.

Класифікація імен за гендерною ознакою є прикладом задачі бінарної класифікації, для вирішення якої достатньою скористатися методом лінійної регресії. Для реалізації цього методу доцільно використати звичайну повнозв’язну нейронну мережу.

1.6. ВИСНОВКИ

Для вирішення проблеми класифікації текстових даних та зображень

доцільно використовувати алгоритми машинного навчання, а саме нейронні мережі. Існують моделі нейронних мереж, які здатні класифікувати різні дані з високою точністю. У роботі для аналізу та класифікації зображень вирішено використовувати глибокі згорткові нейронні мережі які лідирують за точністю класифікації на вибірці даних ImageNet: InceptionResNetV2, Xception, ResNet50. Такі мережі допомагають виділити певні абстракції та ознаки зображень, за якими можна класифікувати об’єкти з певною точністю.

Для аналізу та класифікації текстових даних, а саме імен користувачів соціальних мереж розглянуто декілька методів: використання словників, алгоритмів машинного навчання та обробку природної мови. У роботі вирішено використовувати алгоритми машинного навчання, а саме повнозв’язні нейронні мережі. Для вирішення задачі класифікації імен за гендерною ознакою достатньо використати метод лінійної регресії, який успішно реалізують такі мережі.

2. НАЛАШТУВАННЯ ТА ВИКОРИСТАННЯ ОБРАНИХ МЕТОДІВ АНАЛІЗУ ДАНИХ

Оглянувши існуючі методи аналізу текстових даних та зображень для вирішення задачі класифікації профілів соціальної мережі Instagram за гендерною ознакою, вирішено використовувати нейронні мережі.

Для класифікації зображень використано моделі мереж, які використовують технологію Inception, а для аналізу тексту використано повнозв’язні неронні мережі.

2.1. КЛАСИФІКАЦІЯ ІМЕН ЗА ГЕНДЕРНОЮ ОЗНАКОЮ

Задача класифікації імен за гендерною ознакою є важливою частиною визначення статі користувачів будь-яких інтернет ресурсів, що може бути корисним у економічних, наукових та технічних сферах. А саме, для пошуку персоналу, який має якості, які потрібні для певних вакансій або для реалізації реклами націленої на уподобання покупців. Для вирішення цієї задачі з високою точністю у роботі використовуються алгоритми машинного навчання. Знайдено ознаки імен, які можуть бути важливими для визначення статі. Досліджено різні структури нейронної мережі та запропоновано ті, які найкраще підходять для вирішення задачі.

Дану задачу можна вирішувати різними способами: методами машинного навчання, використанням словників та перевіркою на повне входження, використанням алгоритмів аналізу природної мови. Використання словників (порівняння з відомими іменами) може використовуватися досить точно, але не можливо зберегти всі існуючі імена, тобто така система буде працювати лише для збережених імен і зовсім не буде працювати для нових імен.

Обробка природних мов - це підрозділ штучного інтелекту, який зосереджується на тому, щоб комп'ютери могли розуміти та обробляти людські мови, наближаючи комп'ютери до розуміння мови людини.

Комп'ютери ще не мають інтуїтивного розуміння природної мови, яку виконує людина. У двох словах, комп'ютер не може читати між рядками. Останні досягнення в області машинного навчання дозволили комп'ютерам робити дуже багато корисних речей з природною мовою. Глибоке навчання дозволило нам написати програми для виконання речей, таких як мовний переклад, семантичне розуміння та підсумовування тексту. Всі ці речі додають реальне значення, що полегшує розуміння та виконання обчислень на великих блоках тексту без ручного зусилля. Процес читання та розуміння мови набагато складніший, ніж це здається на перший погляд. Є багато речей, які використовуються, щоб по-справжньому зрозуміти, що означає фрагмент тексту в реальному світі. Технологія обробки природних мов поєднує алгоритми комп'ютерного навчання з обчислювальною лінгвістикою та інформатикою для обробки людських або природних мов та мови. Процес можна розбити на три частини. Перше завдання полягає в розумінні природної мови, отриманої комп'ютером. Комп'ютер використовує вбудовану статистичну модель для виконання процедури розпізнавання мовлення, яка перетворює природну мову на мову програмування. Це робиться, розбиваючи мову, яку він чує в крихітних одиницях, а потім порівнює ці одиниці з попередніми одиницями з попередньої промови. Висновок або результат в текстовому форматі статистично визначає слова та речення, які, швидше за все, сказали. Це перше завдання називається процесом мовлення до тексту.

Наступне завдання називається тегування частин мовлення або двозначне визначення категорій слова. Цей процес елементарно ідентифікує слова в їх граматичних формах як іменники, дієслова, прикметники, тощо, використовуючи набір правил лексики, закодованих в комп'ютер. Після цих двох процесів комп'ютер, розуміє значення висловлюваної мови.

Третій крок - перетворення тексту в мову. На цьому етапі мова комп'ютера програмування перетворюється в звуковий або текстовий формат для користувача. Дана технологія намагається зробити комп'ютери

інтелектуальними, роблячи людей, які вважають, що вони взаємодіють з іншою людиною. Тест Тьюрінга, запропонований Аланом Тьюрингом в 1950 році, говорить, що комп'ютер може бути цілком розумним, якщо він може думати і робити бесіду як людина без людини, знаючи, що він або вона спілкується з машиною. Поки що лише один комп'ютер пройшов тест - чат з персонажем 13-річного хлопчика. Це не означає, що інтелектуальну машину неможливо побудувати, але вона окреслює труднощі, пов'язані з тим, щоб зробити комп'ютер думати або спілкуватися як людина. Оскільки слова можуть бути використані в різних контекстах, і машини не мають реального життєвого досвіду, який люди мають для передачі та опису сутностей зі слів, це може зайняти трохи більше часу, перш ніж світ може повністю відмовитися від мови комп'ютерного програмування.

З опису методів можна побачити, що в даному випадку аналізуються поєднання слів: речення та цілі тексти, що не зовсім підходить для вирішення задачі класифікації імен. Тому для вирішення задачі класифікації імен за гендерною ознакою вирішено використовувати нейронні мережі.

Нейронні мережі - це програмна імплементація нейронних структур людського мозку. Мережа складається з нейронів (кожен з яких приймає зважений вхід, активує *функцію* для суми входів і генерує вихід) та навчається шляхом передавання відповідної вхідної інформації та генерує результат на виході з певною точністю . Структури таких мереж можуть приймати багато різних форм, але найпоширеніша складається з вхідного, прихованого та вихідного шарів. За допомогою таких мереж можливо вирішувати задачі класифікації з високою точністю. Проблема, яка може виникнути під час використання нейронної мережі – це ефект перенавчання, що призводить до адаптації моделі тільки до навчальної вибірки, тобто відбувається просте запам’ятовування, а не виділення корисних ознак, які впливають на успішність класифікації. Для запобігання перенавчання використовується метод виключення (Dropout). Мережі для навчання модифікуються за допомогою виключення з мережі нейронів з

ймовірністю p. Таким чином, ймовірність того, що нейрон залишиться в мережі, становить q = 1 - p. Виключення нейрона означає, що при будь-яких вхідних даних або параметрах він повертає 0. Виключені нейрони не вносять свій внесок в процес навчання на жодному з етапів алгоритму навчання, тому виключення хоча б одного з нейронів рівносильне навчанню нової нейронної мережі.

Вхідними даними для нейронної мережі є набір ознак об'єкта, який класифікується, тому важливо виділити ознаки, які лежатимуть в основі правильної класифікації об'єкта в межах своїх статистичних особливостей. Для навчання нейронних мереж потрібно мати вибірку навчальних даних, яка має інформацію про класи, які потрібно розпізнавати. Було використано базу даних імен новонароджених дітей США, яка налічує більше ніж 95000 імен, які зустрічалися принаймні 5 разів. Дані зберігаються у алфавітному порядку, тому для збільшення ефективності перед кожним навчанням дані були відсортовані у випадковому порядку. Для збільшення швидкодії під час пошуку потрібних ознак імен та структури мережі було використано не всі данні, а лише 1000 випадкових імен, що давало можливість швидко навчати мережу та дивитися на зміни точності класифікації. Процес навчання ділиться на епохи, де кожна епоха відповідає подачі на вхід мережі всіх тренувальних даних. Перед навчанням дані діляться на 3 частини: тестові, дані для перевірки та дані для навчання. Дані для перевірки складають 20% від усіх даних та використовуються в кінці кожної епохи для підрахунку точності під час навчання. Тестова вибірка складається з даних, з якими мережа ніколи не працювала та використовується після навчання для визначення остаточної точності класифікації. Такі дані складають 10% від усіх.

Одним з найважливіших факторів від якого залежить успішність використання обраного методу є правильний підбір вхідних ознак. В даному випадку потрібно знайти такі ознаки імен, за якими можна дійсно класифікувати їх як жіночі або чоловічі. Для імен це зробити досить складно, адже гендерні ознаки імен відрізняються для різних культур та

національностей. Наприклад, слов'янські жіночі імена частіше закінчуються на голосну букву ніж чоловічі, чого точно не можна сказати про імена інших національностей.

На початку знаходження правильних ознак використовується проста нейронна мережа, яка складається з вхідного шару, одного прихованого з 512 нейронів та вихідного з 2 нейронів. Навчання мережі відбувалося протягом 10 епох з використанням 1000 випадкових імен. Перша ознака, яку було використано – частота літер в імені. Перед тренуванням всі імена перетворювалися в масиви з 26 елементів (в англійському алфавіті 26 літер), в якому кожен елемент відповідав частоті появи певної літери в імені, відповідно вхідний шар мережі мав 26 нейронів. Використання цієї ознаки дало точність класифікації в 68%. Наступною розглянутою ознакою був порядок літер в імені. До вхідного масиву додалися ще 26 елементів, які відображали порядок літер, таким чином масив та вхідний шар мережі стали складатися з 56 елементів. Розглянуті ознаки показали точність в 71%. Наступною ознакою, яка була використана стали біграми (2-грами). Біграми – це частини слова по 2 літери, наприклад слово "аабв" складається з біграм "аа", "аб" та "бв". Всього в англійській мові можливо створити 676 біграм, тому вхідний масив став складатися з 732 елементів. Описані ознаки показали точність в 75%. Наступними ознаками були остання літера, передостання літера та довжина слова. Ці ознаки зробили найбільший внесок у досягненні позитивного результату при вирішенні цієї задачі. Після тренування мережі лише на одній ознаці останньої літери, було отримано точність в 75%. Описані вище ознаки разом дали точність в 79%. Останньою використаною ознакою була кількість голосних літер. Ця ознака за своїм змістом схожа на частоту появи літер, але її використання збільшило точність класифікації до 81%.

Під час знаходження оптимальної структури моделі нейронної мережі було проведено багато модифікацій різноманітного характеру. Було перевірено різну кількість прихованих шарів від 1 до 6. Різну кількість нейронів в кожному з шарів від 8192 до 32. Застосовано dropout до кожного

з шарів з різною ймовірністю видалення нейронів. Деякі конфігурації мережі показували високий відсоток точності, але показник перенавчання також був високий, що свідчило про просте запам'ятовування тренувальної вибірки. В результаті експериментів було знайдено конфігурацію, яка показала найвищу точність в 84%: 6 прихованих шарів з 4086, 2048, 1024, 512, 64, 32 нейронів відповідно, до кожного з яких застосований dropout з ймовірністю в 30%. Після тренування на всіх даних (більше ніж 95000 імен) протягом 20 епох було отримано точність класифікації імен за гендерною ознакою в 90%.

Запропоновано використання методів машинного навчання для вирішення задачі класифікації імен за гендерною ознакою. Визначено важливі ознаки імен для даної класифікації та сформовано структуру нейронної мережі, яка показує високу точність класифікації. В ході дослідження виявлено, що важливими ознаками є частота і порядок літер, біграми, остання та передостання літери, довжина слова та кількість голосних літер. Використано наступну структуру мережі: 6 прихованих шарів з 4086, 2048, 1024, 512, 64, 32 нейронів відповідно, до кожного з яких застосований dropout з ймовірністю в 30%.

В результаті дослідження вдалося отримати точніс класифікації в 90%.

Отже, для вирішення задачі класифікації імен за гендерною особливістю використання методів машинного навчання є доцільним та ефективним способом досягнення високої точності класифікації.

2.2. КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ

Основними даними соціальної мережі Instagram є фотографії. Для вирішення задачі визначення статі користувачів цієї мережі дуже важливо точно навчитися класифікувати фотографії. За вхідні дані взяті головні фотографії профілів, та декілька перших фотографій з сторінки (до 10).

У роботі використовувався метод навчання з учителем. Для реалізації цього методу потрібна велика кількість навчальних даних, тобто окремі фотографій на яких присутні чоловіки та окремі на яких є жінки. У роботі дані для навчання завантажено з самої соціальної мережі Instagram. Обрано декілька хештегів, які притаманні жінкам та чоловікам та знайдено приблизно по 3500 профілів кожного класу. Далі з цих профілів завантажено по 30000 фотографій кожної категорії. Зараз ми отримали досить зашумлену вибірку даних. В цій вибірці були фотографії де були присутні декілька людей або не було людей взагалі, або фотографії були мультиплікаційні. Для очищення вибірки використовувалась технологія openCV (реалізація технології комп'ютерного бачення).

2.2.1. КОМП'ЮТЕРНЕ БАЧЕННЯ

Комп'ютерне бачення є міждисциплінарною галуззю, яка стосується того, як комп'ютери можуть бути зроблені, щоб отримати високий рівень розуміння з цифрових зображень або відео. З точки зору техніки, вона прагне автоматизувати завдання, які може зробити людська зорова система. Завдання комп'ютерного зору включають методи отримання, обробки, аналізу та розуміння цифрових зображень та вилучення великогабаритних даних з реального світу з метою отримання числової чи символічної інформації. Розуміння в цьому контексті означає перетворення візуальних зображень (введення сітківки) в описи світу, які можуть взаємодіяти з іншими процесами мислення та виявляти відповідні дії. Це розуміння зображення можна розглядати як розкидання символічної інформації з даних зображення за допомогою моделей, побудованих за допомогою геометрії, фізики, статистики та теорії навчання.

Як наукова дисципліна, комп'ютерне бачення стосується теорії за штучними системами, які витягують інформацію з зображень. Дані зображення можуть мати різні форми, такі як відеопослідовності, перегляди з кількох камер або багатовимірні дані з медичного сканера. Як

технологічна дисципліна, комп'ютерне бачення прагне застосовувати свої теорії та моделі для побудови систем комп'ютерного зору. Технології комп'ютерного бачення включають реконструкцію сцени, виявлення подій, відстеження відео, розпізнавання об'єктів, оцінку 3D-пори, навчання, індексацію, оцінку руху та відновлення зображення.

Комп'ютерне бачення є міждисциплінарною галуззю, яка стосується того, як комп'ютери можуть бути зроблені, щоб отримати високий рівень розуміння з цифрових зображень або відео. З точки зору техніки, вона прагне автоматизувати завдання, які може зробити людська зорова система. Комп'ютерне бачення стосується автоматичного вилучення, аналізу та розуміння корисної інформації з одного зображення або послідовності зображень, що передбачає розробку теоретичної та алгоритмічної основи для досягнення автоматичного візуального розуміння. Як наукова дисципліна, комп'ютерне бачення стосується теорії за штучними системами, які витягують інформацію з зображень. Дані зображення можуть мати різні форми, такі як відеопослідовність, перегляди з кількох камер або багатомірні дані з медичного сканера. Як технологічна дисципліна, комп'ютерне бачення прагне застосовувати свої теорії та моделі для побудови систем комп'ютерного зору.

Області штучного інтелекту стосуються автономного планування або обговорення робототехнічних систем для навігації по навколишньому середовищу. Детальне розуміння цих середовищ необхідне для навігації по них. Інформація про навколишнє середовище може бути забезпечена системою комп'ютерного зору, яка діє як датчик зору і забезпечує високоякісну інформацію про навколишнє середовище та робот.

Штучний інтелект та комп'ютерне бачення поділяють інші теми, такі як розпізнавання образів та методи навчання. Отже, комп'ютерне бачення іноді розглядається як частина галузі штучного інтелекту або галузі інформатики в цілому.

Фізика твердого тіла - це інша галузь, яка тісно пов'язана з комп'ютерним баченням. Більшість систем комп'ютерного зору спираються на датчики

зображення, які виявляють електромагнітне випромінювання, яке, як правило, має вигляд або інфрачервоного світла. Датчики розроблені з використанням квантової фізики. Процес, за допомогою якого світло взаємодіє з поверхнями, пояснюється фізикою. Фізика пояснює поведінку оптики, яка є основною частиною більшості систем візуалізації. Досконалі сенсори зображення навіть вимагають квантової механіки, щоб забезпечити повне уявлення про процес формування зображення. Також різні задачі вимірювання в фізиці можна вирішити, використовуючи комп'ютерне бачення, наприклад рух у рідинах.

Третя область, що відіграє важливу роль, - нейробіологія, зокрема вивчення системи біологічного бачення. Протягом останнього сторіччя було проведено широке вивчення очей, нейронів та структур мозку, присвячених обробці візуальних стимулів як у людини, так і у різних тварин. Це призвело до грубого, але складного опису того, як діють "реальні" системи зору, щоб вирішити певні завдання, пов'язані з баченням. Ці результати призвели до підполя в комп'ютерному баченні, де штучні системи призначені для імітації обробки та поведінки біологічних систем на різних рівнях складності. Крім того, деякі з методів навчання, розроблених в межах комп'ютерного бачення (наприклад, нейронна мережа та аналіз і класифікація іміджу та функцій, що базуються на глибокому вивченні), мають біологічну основу.

Деякі напрями досліджень комп'ютерного зору тісно пов'язані з вивченням біологічного бачення - дійсно, так само, як багато напрямків досліджень AI тісно пов'язані з дослідженнями людської свідомості, а також використанням накопичених знань для інтерпретації, інтеграції та використання візуальної інформації. Поле біологічного зору досліджує і моделює фізіологічні процеси, що стоять позаду зорового сприйняття у людей та інших тварин. З іншого боку, комп'ютерне бачення вивчає і описує процеси, що реалізуються в програмному та апаратному обладнанні за системами штучного зору. Міждисциплінарний обмін між біологічним та комп'ютерним баченням виявився плідним для обох областей.

Ще одна область, пов'язана з комп'ютерним баченням, - обробка сигналів. Багато методів обробки одномірних сигналів, як правило, тимчасових сигналів, можуть природним чином поширюватися на обробку двох змінних сигналів або мультиперемінних сигналів для комп'ютерного зору. Проте в силу специфіки зображення існує безліч методів, розроблених в рамках комп'ютерного зору, які не мають аналогів при обробці одномірних сигналів. Разом з багатомірністю сигналу, це визначає підполе обробки сигналу як частину комп'ютерного бачення.

Окрім вищезгаданих поглядів на комп'ютерне бачення, багато з відповідних тем дослідження також можуть бути вивчені з суто математичної точки зору. Наприклад, багато методів комп'ютерного бачення базуються на статистиці, оптимізації чи геометрії. Нарешті, значна частина поля присвячена аспекту реалізації комп'ютерного бачення; як існуючі методи можуть бути реалізовані в різних комбінаціях програмного та апаратного забезпечення, або як ці способи можуть бути змінені, щоб отримати швидкість обробки, не втрачаючи занадто високу продуктивність.

Найбільш тісно пов'язані з комп'ютерним баченням проблеми - обробка зображень, аналіз зображень та машинного бачення. Існує значний збіг у спектрі методів та програм, які вони охоплюють. Це означає, що основні методи, які використовуються та розробляються на цих полях, подібні, що можна інтерпретувати, оскільки існує лише одне поле з різними назвами. З іншого боку, дослідницьким групам, науковим журналам, конференціям та компаніям видається необхідним представляти або продавати себе як такі, що належать конкретно до однієї з цих областей, і, отже, різноманітні характеристики, що відрізняють кожне з полів від інших, були представлені.

Комп'ютерна графіка створює дані зображень з 3D-моделей, комп'ютерне бачення часто виробляє 3D-моделі з даних зображення. Існує також тенденція до поєднання двох дисциплін, наприклад, як дослідженню в розширеній реальності.

Наступні характеристики є актуальними, але їх не слід вважати загальноприйнятими.

Обробка та аналіз зображень, як правило, зосереджуються на 2D-зображеннях, як перетворити одне зображення на інше, наприклад, піксельними операціями, такими як підвищення контрастності, локальні операції, такі як видалення краю або видалення шумів, або геометричні перетворення, такі як обертання зображення . Ця характеристика означає, що обробка / аналіз зображення не потребують припущень і не дають інтерпретацій щодо вмісту зображення.

Комп'ютерне бачення включає 3D-аналіз з 2D-зображень. Аналізує 3D-сцену, яка проеціюється на одне або декілька зображень, наприклад, як відновити структуру або іншу інформацію про 3D-сцену з одного або декількох зображень. Комп'ютерне бачення часто спирається на більш-менш складні припущення про сцену.

Машинний зріз - це процес застосування різноманітних технологій та методів для автоматизованої інспекції, керування процесом та керування роботами на промислових пристроях. Машина зору, як правило, зосереджується на застосуванні, головним чином у виробництві, наприклад, на базі зору на базі роботів та систем для огляду, вимірювання чи вибору, що базуються на зору (наприклад, збирання сміття). З цього випливає, що технології сенсорів зображень та теорія управління часто інтегровані з обробкою даних зображення для керування роботом, а процес обробки в режимі реального часу підкреслюється за допомогою ефективної реалізації в апаратному та програмному забезпеченні. Це також означає, що зовнішні умови, такі як освітлення, можуть бути і часто контролюються у машинному зору, ніж у загальному комп'ютерному баченні, що може дозволити використання різних алгоритмів.

Існує також розділ, який називається візуалізацією, яке перш за все зосереджується на процесі створення зображень, але іноді також займається обробкою та аналізом зображень. Наприклад, медична візуалізація включає значну роботу з аналізу даних зображення в медичних програмах.

Нарешті, розпізнавання моделей - це технологія, яка використовує різні методи для отримання інформації від сигналів в цілому, переважно на основі статистичних підходів та штучних нейронних мереж. Значна частина цього поля присвячена застосуванню цих методів до даних зображення.

Фотограмметрія також збігається з комп'ютерним баченням, наприклад, стереофотограмметрією або комп'ютерним стереоефектом.

Області застосування таких завдань, як системи промислового машинного зору, які, скажімо, перевіряють швидкість завантаження пляшок на виробничій лінії, дослідження штучного інтелекту та комп'ютерів або роботів, які можуть осмислити навколишній світ. Поля комп'ютерного зору та області машинного бачення значно збігаються. Комп'ютерне бачення охоплює базову технологію автоматизованого аналізу зображень, яка використовується у багатьох областях. Машинного зору, як правило, називається процес об'єднання автоматизованого аналізу зображень з іншими методами та технологіями для забезпечення автоматизованого інспектування та керування роботами в промислових цілях. У багатьох програмах для перегляду програм комп'ютери попередньо запрограмовані для вирішення певного завдання, але методи, засновані на навчанні, стають все більш поширеними. Приклади застосування комп'ютерного бачення включають системи для:

- Автоматична перевірка;
- Надання допомоги людям в ідентифікаційних завданнях;
- Контрольні процеси, наприклад, промисловий робот;
- Виявлення подій, наприклад, для візуального спостереження або підрахунку людей;
- Взаємодія, наприклад, як вхідний пристрій для взаємодії комп'ютера та людини;
- Моделювання об'єктів або середовищ, наприклад, медичний аналіз зображень або топографічне моделювання;

- Навігація, наприклад, автономним транспортним засобом або мобільним роботам;
- Організація інформації, наприклад, для індексації баз даних зображень та послідовностей зображень.

Як вже згадувалося у роботі використовувалася бібліотека openCV, за допомогою якої реалізовано очищення зображень з тестового набору даних. Видалялися ті фотографії, на яких немає облич, тобто скоріш за все, якщо на фотографії немає обличчя, на ній не має людини. Таким чином ми отрімали приблизно по 10000 фотографій кожної категорії для навчання.

2.2.2. ВИБІР МОДЕЛІ НЕЙРОННОЇ МЕРЕЖІ

У роботі розглянуто декілька готових моделей нейронних мереж (табл. 1).

Таблиця 1- Характеристики відомих моделей нейронних мереж

Назва	Розмір, Мб	Топ-1 точність	Топ-5 точність	К-сть параметрів	Глибина
Xception	88	0.790	0.945	22,910,480	126
VGG16	528	0.713	0.901	138,357,544	23
VGG19	549	0.713	0.900	143,667,240	26
ResNet50	99	0.749	0.921	25,636,712	168
InceptionV3	92	0.779	0.937	23,851,784	159
InceptionResNetV2	215	0.803	0.953	55,873,736	572
MobileNet	16	0.704	0.895	4,253,864	88

Назва	Розмір, Мб	Топ-1 точність	Топ-5 точність	К-сть параметрів	Глибина
MobileNetV2	14	0.713	0.901	3,538,984	88
DenseNet121	33	0.750	0.923	8,062,504	121
DenseNet169	57	0.762	0.932	14,307,880	169
DenseNet201	80	0.773	0.936	20,242,984	201
NASNetMobile	23	0.744	0.919	5,326,716	-
NASNetLarge	343	0.825	0.960	88,949,818	-

Розглянуті моделі розробили під час змагань з класифікації зображень на базі даних ImageNet, яка налічую більше 1000000 зображень, з яких було обрано 1000 класів для класифікації. Колонка таблиці, яка має назву “Топ-1 точності” показує відсоток правильно класифікованих зображень для кожної моделі. Кількість параметрів показує кількість ваг, які налаштовуються під час навчання. Беручи до уваги кількість параметрів та розміри ваг, можна зробити висновки щодо швидкості роботи кожної мережі. Зважаючи на те, що у роботі потрібно аналізувати велику кількість даних стає зрозуміло, що швидкість обробки має значення, тому для використання у роботі обрано “легку” мережу MobileNet. Дана модель займає не багато пам'яті (16 Мб) та може працювати на пристроях з обмеженими ресурсами. Для навчання такої мережі можливо використовувати один із декількох можливих алгоритмів. Розглянемо кожний з них детальніше.

2.2.3. АЛГОРИТМИ НАВЧАННЯ

Алгоритми оптимізації допомагають нам мінімізувати (або максимізувати) функцію помилки E (x), яка є просто математичною функцією, яка залежить від внутрішніх параметрів, які вивчає модель, які використовуються для обчислення цільових значень (Y) з набору предикторів (X), що використовуються в моделі. Наприклад, ми називаємо ваги (W) та значення Bias (b) нейронної мережі як внутрішні параметри, які можна вивчати, які використовуються при обчисленні вихідних значень, а також вивчаються та оновлюються в напрямку оптимального рішення, тобто мінімізація втрат - процес навчання мережі, а також відіграють важливу роль у процесі навчання нейронної мережі. Внутрішні параметри моделі відіграють дуже важливу роль у ефективному навчанні моделі та забезпечують точні результати. Ось чому ми використовуємо різні стратегії та алгоритми оптимізації для оновлення та обчислення відповідних та оптимальних значень параметрів такої моделі, які впливають на навчальний процес нашої моделі.

Оптимізаційні алгоритми поділяються на дві категорії: алгоритми оптимізації першого та другого порядку. Алгоритми оптимізації першого порядку мінімізують або максимізують функцію втрат E (x), використовуючи свої значення градієнта по відношенню до параметрів. Найпоширеніший алгоритм оптимізації першого порядку - це алгоритм градієнтного спуску. Похідна першого порядку говорить нам, чи зменшується чи збільшується функція в певній точці. Первинний порядок похідних в основному дає нам лінію, яка є тангенціальною до точки на площину помилки. Градієнт - це вектор, який є узагальненням похідної (dy / dx), що є миттєвою швидкістю зміни у по відношенню до x. Різниця полягає в тому, що для розрахунку похідної функції, яка залежить від більш ніж однієї змінної або декількох змінних, градієнт займає його місце. Градієнт розраховується за допомогою часткових похідних. Ще однією серйозною відмінністю між градієнтом і похідною є те, що градієнт функції створює векторне поле.

Алгоритми оптимізації другого порядку - використовують похідну другого порядку, яка також називається гессіан, щоб звести до мінімуму або максимізувати функцію втрат. Гессіан є матрицею часткових похідних другого порядку. Оскільки друга похідна дорого обчислюється, другий порядок не використовується значно. Похідна другого порядку говорить нам, чи збільшується чи зменшується перша похідна, що натякає на кривизну функції. Другий порядок похідних дає нам квадратичну поверхню, яка торкається кривизни поверхні помилки.

Зараз методи оптимізації першого замовлення легко обчислюються та забирають менше часу, швидко зближуючись на великих наборах даних. Техніки другого порядку швидше, лише якщо похідна другого порядку відома, ці методи завжди більш повільні та дорогі для обчислення в термінах як часу, так і пам'яті.

Градієнтне сходження є найважливішим методом і основою того, як ми тренуємо та оптимізуємо інтелектуальні системи. $\theta = \theta - \eta \cdot \nabla J(\theta)$ - це формула оновлення параметрів, де 'η' є швидкістю навчання, '$\nabla J(\theta)$' - функція втрати градієнта. Це найпопулярніші алгоритми оптимізації, що використовуються для оптимізації нейронних мереж. Тепер градієнтний спуск використовується для того, щоб робити оновлення ваг в моделі нейромережі, тобто оновити та налаштовувати параметри моделі в напрямку, щоб ми могли мінімізувати функцію втрат. Нейромережі використовують техніку, яка називається зворотнє поширення помилки, в якій ми спочатку поширюємо передові обчислення точкового продукту сигналів вводу та їх відповідних ваг, а потім застосовуємо функцію активації до цієї суми елементів, яка перетворює вхідний сигнал на вихідний сигнал, а також важливий для моделювання складних нелінійних функцій і вводить нелінійну модель, яка дозволяє моделі вивчати практично будь-які довільні функціональні відображення. Після цього ми поширюємо назад у мережу, що містить умови помилки та оновлення значень ваг з використанням градієнтного спуску, в якому ми обчислюємо градієнт

функції Error (E) щодо маси (W) або параметрів та оновлюємо параметри у протилежному напрямку функції , що відповідає параметрам моделі. Традиційний градієнтний спуск порахує градієнт всього набору даних, але виконуватиме лише одне оновлення, отже дуже повільний і важко контролювати наборами даних, які дуже великі. Наскільки велика чи мала кількість оновлень, яка гарантовано сходиться до глобального мінімуму для опуклих поверхонь помилок і до локального мінімуму для неопуклих поверхонь. Вищезазначені проблеми стандартного градієнтного спуску виправляються в стохастичному градієнтному спуску.

Стохастичний градієнтний спуск (SGD), з іншого боку, виконує оновлення параметрів для кожного навчального прикладу. Зазвичай це набагато швидше. Тепер завдяки цим частішим оновленням параметри мають високу дисперсію і це призводить до того, що функція втрат коливається до різних інтенсивностей. Це дійсно добре, оскільки це допомагає виявити нові та, можливо, кращі локальні мінімуми, тоді як стандартний алгоритм буде сходитися лише до одного мінімуму, як було зазначено вище. Але проблема з SGD полягає в тому, що через високу частоту оновлення в кінцевому підсумку ускладнює конвергенцію до точного мінімуму і продовжуватиме коливатися. Проблеми з оновленнями параметрів високої дисперсії та нестабільною конвергенції можуть бути виправлені в іншому варіанті, званому Mini-Batch Gradient Descent (мініатюрний градієнтний спуск). Удосконалення, щоб уникнути всіх проблем і недоліків SGD та стандартного спуску, полягає у використанні мініатюрного градієнтного спуску, оскільки він використовує найкраще з обох методів і виконує оновлення для кожної партії з наборами навчальних прикладів в кожній партії. Однією з переваг такого методу є зменшення дисперсії в оновлених параметрах, що, зрештою, приведе до кращої та стабільної конвергенції. Притаманною ознакою для методу є розміри партій в діапазоні від 50 до 256, але можуть відрізнятися залежно від застосування та вирішення проблеми. Вибір правильного курсу навчання може бути складним. Швидкість навчання, яка є занадто мала, призводить до

хворобливого повільного зближення, тобто призведе до того, що маленькі кроки знайдуть оптимальні значення параметрів, які мінімізують втрати та впливає на загальний час навчання, який надто великий. Хоча швидкість навчання, яка занадто велика, може заважати конвергенції та призводити до того, що функція втрат коливається навколо мінімуму або навіть розходиться. Крім того, однаковий рівень навчання застосовується до всіх оновлених параметрів. Ще однією ключовою проблемою мінімізації нерівномірних функцій помилок, загальних для нейронних мереж, є запобігання потрапляння в пастку їх численних недооптимальних локальних мінімумів. Власне, складність виникає не з місцевих мінімумів, а від точок сідла, тобто точок, де один вимір схиляється, а інший схиляється вниз. Ці точки сідла, як правило, оточені плоскогір'ями тієї самої помилки, що робить його надзвичайно важким для виходу SGD, оскільки градієнт близький до нуля у всіх вимірах.

Високі дисперсні коливання в SGD ускладнюють досягнення зближення, тому винайдено техніку імпульсу, яка прискорює SGD шляхом навігації по відповідному напрямку і пом'якшує коливання в невідповідних напрямках. Іншими словами, все це додає фракцію "γ" вектора оновлення останнього кроку до поточного вектора оновлення. Термін імпульсу γ зазвичай встановлюється до 0.9 або подібного значення. Тут імпульс такий же, як імпульс класичної фізики, коли ми кидаємо м'яч вниз з пагорба, він набирає імпульс і його швидкість продовжує зростати. Термін імпульсу γ збільшується для розмірів, градієнти яких вказують в однакових напрямках і зменшують оновлення для розмірів, градієнти яких змінюють напрямки. Це означає, що параметри оновлюються лише для відповідних прикладів. Це зменшує необов'язкові оновлення параметрів, що призводить до більш швидкої та стабільної конвергенції та зменшення коливань.

Дослідник Юрій Нестеров побачив проблему пов’язану з імпульсом - м'яч, який котиться з пагорба, сліпо спускається по схилу, що є дуже незадовільним. Те, що насправді відбувається, полягає в тому, що, коли ми досягаємо мінімумів, тобто найнижчої точки на кривій, імпульс досить

високий, і він не може сповільнитись в цьому місці, що може повністю спричинити пропуск мінімумів. Юрій Нестеров опублікував статтю у 1983 році, яка вирішила цю проблему і ми зараз називаємо цю стратегію нестримовим прискореним градієнтом.

У запропонованому методі він запропонував нам спочатку зробити великий стрибок на основі попереднього імпульсу, потім обчислити градієнт і зробити їх корекцію, в результаті якої буде оновлено параметр. Тепер це попереднє оновлення не дозволяє нам зайти надто швидко, не пропустити мінімуми та робити його більш чутливими до змін. Тепер, коли ми можемо адаптувати наші оновлення до нахилу нашої функції помилок і прискорити швидкість SGD, ми також хотіли б адаптувати наші оновлення до кожного окремого параметра, щоб виконати більші або менші оновлення залежно від їх важливості.

Оскільки ми розраховуємо індивідуальні курси навчання для кожного параметра, то чому б не обчислити індивідуальні зміни імпульсів для кожного параметра і зберігати їх окремо. Саме тут починається нова модифікація техніки та вдосконалення, яку називають Адамом (Adam). Адам стоїть за оцінкою адаптивного моменту. Оцінка адаптивного моменту (Adam) - це ще один метод, який обчислює адаптивні темпи навчання для кожного параметра. Окрім зберігання експоненціально затухаючої середньої кількості квадратних градієнтів, таких як AdaDelta, Адам також зберігає експоненціально падаюче середнє значення минулих градієнтів M (t), подібних до імпульсу: M (t) і V (t) - значення першої моменту, яка є середньою, а друга моментом - нецентрованою дисперсією градієнтів відповідно. Адам добре працює на практиці і вигідно відрізняється від інших алгоритмів адаптивного алгоритму навчання, оскільки він дуже швидко збігається, а також виправляє всі проблеми, що виникають в інших методах оптимізації, таких як зникнення швидкості навчання , повільна конвергенція або висока дисперсія в оновленні параметрів, що призводить до коливання функції втрат.

2.2.4. МЕТОДИ ПОКРАЩЕННЯ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

Розглянуті методи включають: кращий вибір функції витрат, відома як функція вартості крос-ентропії; чотири так званих "регуляризаційних" методи (L1 і L2 регуляризація, виключення та штучне розширення тренувальних даних), що робить наші мережі більш узагальненими за межами навчальних даних; кращий спосіб ініціалізації ваг у мережі; і набір евристики, щоб допомогти вибрати хороші гіперпараметри для мережі. Також розглядаються кілька інших методів. Звичайно, ми розглядаємо лише деякі з численних багатьох методів, розроблених для використання в нейронних мережах. Філософія полягає в тому, що найкращим внеском в безліч доступних прийомів є поглиблене вивчення кількох найважливіших. Освоєння цих важливих прийомів не тільки корисно для вирішення поставленої задачі, але також поглиблює розуміння того, які проблеми можуть виникнути при використанні нейронних мереж.

Розглянемо нейрон лише з одним входом. Ми навчимо цей нейрон робити щось дуже просте: на вході 1, на виході 0. Звичайно, це настільки тривіальне завдання, що ми могли б легко розібрати відповідну вагу та зміщення вручну, не використовуючи алгоритм навчання. Тим не менш, він, очевидно, освітлює використання градієнтного спуску, щоб спробувати вчити ваги та упередженості. Тож давайте поглянемо, як нейрон навчається. Для того, щоб зробити речі певними, виберемо початкову вагу 0,6 та початкове зміщення 0,9. Це загальний вибір, який використовується для початку навчання, це не важливо в будь-якому випадку. Початковий вихід нейрона дорівнює 0,82, тому знадобиться трохи навчання, перш ніж наш нейрон досягне бажаного результату, 0,0. Швидкість навчання $\eta = 0,15$, яка виявляється досить повільною, що ми можемо слідкувати за тим, що відбувається, але досить швидко, щоб ми могли отримати значне навчання всього за кілька секунд. Вартість - це функція квадратичної вартості, C.

Як видно, нейрон швидко змінює вагу та зміщення, що знижує вартість, і дає вихід з нейрона близько 0,09. Це не зовсім бажаний результат, 0.0, але це досить добре. Припустимо, однак, що ми замість цього виберемо початкові ваги та початковий ухил 2.0. У цьому випадку на виході отримуємо 0.98, що дуже неправильне. Подивимося, як нейрон навчається у цьому випадку. Хоча в цьому прикладі використовується однакова швидкість навчання (η = 0.15), ми бачимо, що навчання починається набагато повільніше. Дійсно, для перших 150 навчальних епох, ваги та упередження не дуже змінюються. Потім навчання починається і, як і в нашому першому прикладі, вихід нейрона швидко рухається ближче до 0.0. Ця поведінка є дивною, що протиставляється навчанню людей. Але ми тільки що побачили, що наш штучний нейрон має багато труднощів у навчанні, ми маємо поганий результат на початку навчання - набагато складніше, ніж тоді, результат трохи відрізняється від бажаного. Більше того, виявляється, така поведінка спостерігається не тільки в цій моделі, але і в більш загальних мережах. Щоб зрозуміти походження проблеми, врахуємо, що наш нейрон вивчається зміною ваги та зміщення зі швидкістю, визначеною частковими похідними функції витрат ∂C / ∂w та ∂C / ∂b. Тож кажучи, що "навчання є повільним", це дійсно те ж саме, що сказати, що ці часткові похідні є малими. Завдання полягає в тому, щоб зрозуміти, чому вони невеликі. Щоб зрозуміти це, давайте обчислимо часткові похідні. З рівняння:

$$C = \frac{(y-a)^2}{2},$$

де a - вихід нейрона, коли використовується тренувальний ввід x = 1, а y = 0 - відповідний бажаний вихід. Щоб написати це більш чітко в термінах ваги та упередженості, нагадаємо, що a = σ (z), де z = wx + b. Використовуючи правило ланцюга, щоб диференціювати по відношенню до ваги та зміщення, ми отримуємо:

$$\frac{\partial C}{\partial w} = (a - y)\sigma'(z)x = a\sigma'(z)$$
$$\frac{\partial C}{\partial b} = (a - y)\sigma'(z) = a\sigma'(z),$$

де x = 1 і y = 0. Щоб зрозуміти поведінку цих рівнянь, давайте уважніше розглянемо термін σ '(z) у правій частині. Нагадаємо форму функції σ (рис. 10.)

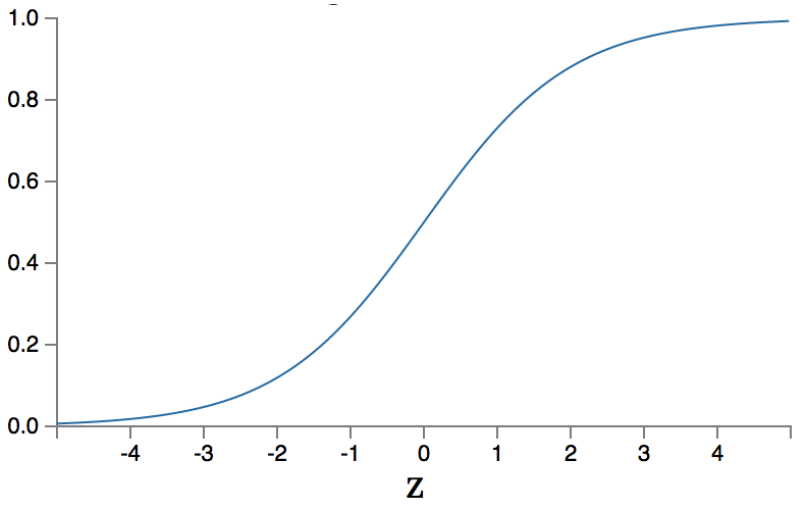

Рисунок 10 - Графік сигмоїдної функції

З цього графіка можна побачити, що коли вихід нейрона близький до 1, крива стає дуже рівною, і тому σ '(z) стає дуже маленьким. Потім розглянуті рівняння говорять нам, що ∂C / ∂w і ∂C / ∂b стають дуже малими. Це породжує уповільнення навчання. Більше того, як ми побачимо трохи пізніше, сповільнення навчання відбувається по суті з тією ж причиною в більш загальних нейронних мережах, а не тільки в наведенному прикладі.

Як ми можемо вирішити проблему уповільнення навчання? Виявляється, ми можемо вирішити проблему, замінивши квадратну вартість з іншою функцією вартості, відома як крос-ентропія. Щоб зрозуміти крос-ентропію, давайте трохи відійдемо від простого наведого прикладу. Замість цього будемо намагатися тренувати нейрон з кількома вхідними змінними, x1, x2, ..., відповідними вагами w1, w2, ..., та

зміщенням, b. Вихід нейрона, звичайно, a = σ (z), де z = Σjwjxj + b - це вагова сума вхідних даних. Визначимо функцію кросс-ентропії для цього нейрона:

$$C = -\frac{1}{n} \sum_x [y \ln a + (1 - y) \ln(1 - a)] ,$$

де n - загальна кількість предметів навчальних даних, сума перевищує всі вхідні дані для тренувань, x та y - відповідний бажаний результат.

Не цілком очевидно, що вираз фіксує проблему уповільнення навчання. Перш ніж звертатися до уповільнення навчання, давайте подивимося, в якому сенсі крос-ентропія може бути інтерпретована як функція вартості. Зокрема, два властивості дозволяють інтерпретувати крос-ентропію як функцію вартості. По-перше, це не негативне значення, тобто C> 0. Щоб це побачити, звернемо увагу, що: всі окремі умови суми у є негативними, оскільки обидва логарифми мають числа в діапазоні від 0 до 1. По-друге, якщо фактичний вихід нейрона близький до бажаного результату для всіх тренувальних входів x, то крос-ентропія буде близька до нуля. Щоб побачити це, припустимо, наприклад, що y = 0 і a≈0 для деякого входу x. Це той випадок, коли нейрон працює над цим входом. Ми бачимо, що перший член в виразі для вартості зникає, оскільки y = 0, а другий - лише -ln (1-a) ≈ 0. Аналогічний аналіз виконується, коли y = 1 і a≈1. Отже, внесок у вартість буде низьким за умови, що фактичний результат близький до бажаного результату. Підводячи підсумок, крос-ентропія позитивна і прагне до нуля, оскільки нейрон покращується при обчисленні бажаного виходу, y для всіх вхідних даних тренування x. Ці обидві властивості, які ми інтуїтивно очікуємо за функцію вартості. Дійсно, обидва властивості також задовольняються квадратичною вартістю. Отже, це гарна новина для крос-ентропії. Але функція витрат на перехресні ентропії має вигоду, що, на відміну від квадратичних витрат, це дозволяє уникнути проблеми навчання уповільнення. Щоб побачити це, давайте обчислимо часткову похідну від витрат крос-ентропії щодо ваг. Ми замінюємо a = σ (z), і застосовуємо правило ланцюга двічі, отримуючи:

$$\frac{\partial C}{\partial w_j} = -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\frac{\partial \sigma}{\partial w_j}$$
$$= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\sigma'(z)x_j.$$

Це говорить нам, що швидкість навчання ваги контролюється σ (z) -y, тобто через помилку у виведенні. Чим більше помилка, тим швидше нейрон навчатиметься. Це просто те, що ми інтуїтивно очікуємо. Зокрема, це дозволяє уникнути уповільнення навчання, викликаного терміном σ '(z) в аналогічному рівнянні для квадратичної вартості. Коли ми використовуємо крос-ентропію, термін σ '(z) скасовується, і нам більше не потрібно турбуватися про те, що він малий. Це забезпечується функцією кросс-ентропії.

Повернемося до прикладу, який ми розглядали раніше, і досліджуємо, що станеться, коли ми використовуємо крос-ентропію замість квадратичної вартості. Щоб знову орієнтуватися, ми почнемо з того випадку, коли квадратичні витрати склали відмінно, з початковою масою 0,6 і вихідним зміщенням 0,9. Не дивно, що в цьому випадку нейрон чудово вчиться, як це було раніше. А тепер давайте подивимося на випадок, коли наш нейрон застряг раніше, з вагою та упередженості, починаючи з 2.0. Цього разу нейрон швидко навчався, як ми сподівалися. Нахил кривої витрат був набагато крутішим спочатку, ніж початкова плоска область на відповідній кривій для квадратичної вартості. Цю крутизну забезпечує крос-ентропія, перешкоджаючи нам застрягти, коли ми сподіваємось, що наш нейрон швидко навчиться, тобто, коли нейрон починає навчатися погано. Раніше, при квадратичній вартості, ми використовували η = 0,15. Чи повинні ми використати ту ж саму швидкість навчання в нових прикладах? Фактично, при зміні функції витрат не можна точно сказати, що означає використовувати "той же" рівень навчання. Для функцій вартості я просто експериментував з пошуком курсу навчання, який давав змогу побачити, що відбувається. Точка графіків полягає не в абсолютній швидкості навчання. Йдеться про те, як змінюється швидкість навчання. Зокрема, коли

ми використовуємо квадратичні витрати, навчання відбувається повільніше, коли нейрон однозначно неправильний, ніж пізніше, оскільки нейрон наближається до правильного виходу; тоді як навчання крос-ентропії швидше, коли нейрон однозначно неправильний. Ми розглянули крос-ентропію для одного нейрона. Тим не менш, легко узагальнити крос-ентропію для багато-нейронних багатошарових мереж. Зокрема, припустимо, що y = y1, y2, ... є бажаними значеннями вихідних нейронів, тобто нейронів у кінцевому шарі, тоді як aL1, aL2, ... є фактичними вихідними значеннями.

$$C = -\frac{1}{n} \sum_x \sum_j \left[y_j \ln a_j^L + (1 - y_j) \ln(1 - a_j^L) \right].$$

Зараз ми отримали Σj підсумовування над усіма вихідними нейронами. До речі, термін "крос-ентропія" використовується таким чином, що може конфліктувати з попередніми термінами, оскільки поверхнево виявляється, що він суперечить іншим джерелам. Зокрема, загальним є визначення крос-ентропії для двох розподілів імовірності, pj і qj. Це визначення може бути пов'язане, якщо ми розглянемо один сигмовидний нейрон як вихідний розподіл ймовірності, що складається з активації нейрона a та її доповнення 1-a. Однак, коли у нас є багато сигмовидних нейронів у кінцевому шарі, вектор aLj активації зазвичай не формує розподіл імовірності. Як результат, визначення навіть не має сенсу, оскільки ми не працюємо з розподілом імовірності. Замість цього, ви можете думати про рівняння як сумарний набір кросс-ентропій для нейронів, причому активація кожного нейрона трактується як частина розподілу ймовірності з двома елементами. У цьому сенсі рівняння є узагальненням крос-ентропії для розподілів імовірності. Коли ми повинні використовувати крос-ентропію замість квадратичної вартості? Фактично, крос-ентропія майже завжди є найкращим вибором, якщо вихідними нейронами є сигмовидні нейрони. Щоб зрозуміти, чому, вважають, що коли ми запускаємо мережу, ми зазвичай ініціалізуємо ваги та упередження, використовуючи якусь рандомізацію. Може статися, що ці вихідні варіанти призводять до того, що мережа є вирішальною мірою

неправильною для деяких вхідних даних - тобто вихідний нейрон буде насичений біля 1, коли воно має бути 0 або навпаки. Якщо ми використовуємо квадратичні витрати, які сповільнять навчання. Це не зупинить навчання повністю, оскільки ваги будуть продовжувати вчитися з інших навчальних матеріалів, але це, очевидно, небажано.

Зараз було в основному використано функцію крос-ентропії для вирішення проблеми уповільнення навчання. Дану проблему можливо вирішувати по іншому, а саме за допомогою softmax шарів нейронів. Ідея softmax - визначити новий тип вихідного шару для наших нейронних мереж. Він починається так само, як і сигмовидний шар, утворюючи зважені входи. Однак, ми не застосовуємо функцію sigmoid для отримання результату. Замість цього в слабкому шарі ми застосовуємо так звану функцію softmax для zLj. Відповідно до цієї функції, активація aLj j-го вихідного нейрона є

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}},$$

де в знаменнику ми знаходимо суму усіх вихідних нейронів.

Одразу не дуже очевидно, що це допоможе нам подолати проблему уповільнення навчання. Щоб краще зрозуміти рівняння, припустімо, що у нас є мережа з чотирма вихідними нейронами та чотирма відповідними зваженими входами, які ми позначимо zL1, zL2, zL3 та zL4. Під час збільшення zL4 відбувається збільшення відповідної активації виходу, aL4 та зменшення інших активних виходів. Точно так само, якщо зменшити zL4, то aL4 зменшиться, а всі інші активації виходу збільшаться. Фактично, в обох випадках загальна зміна інших активацій точно компенсує зміну aL4. Причиною є те, що вихідні активації гарантуються завжди сумою 1, як ми можемо довести, використовуючи рівняння. В результаті, якщо aL4 збільшується, то інші активації виходу повинні зменшуватися на ту саму загальну суму, щоб забезпечити, що сума за всіма активаціями залишається 1. І, звичайно, подібні твердження зберігаються для всіх інших активацій.

Останнє рівняння також означає, що активації виходу є позитивними, оскільки експоненціальна функція є позитивною. Поєднуючи це з спостереженням, ми бачимо, що вихід із рівня softmax являє собою набір позитивних чисел, які підсумовують до 1. Іншими словами, висновок із softmax-шару можна розглядати як розподіл імовірності.

Той факт, що рівень слабкого максимуму виводить розподіл імовірності, досить приємний. У багатьох проблемах зручно тлумачити вихідну активацію aLj як оцінку мережі ймовірності того, що правильний вихід j. Так, наприклад, у проблемі класифікації MNIST ми можемо інтерпретувати aLj як оціночну ймовірність мережі, що правильна класифікація цифр j. На відміну від цього, якщо вихідний шар був сигмовим шаром, то, звичайно, ми не могли припустити, що активації сформували розподіл імовірності. Активація із сигмовидного шару взагалі не буде розподілом імовірності. І так з сигмовим вихідним шаром ми не маємо такої простої інтерпретації вихідних активацій.

2.2.5. ПЕРЕНАВЧАННЯ ТА РЕГУЛЯРИЗАЦІЯ

Моделі з великою кількістю вільних параметрів можуть описувати вражаюче широкий спектр явищ. Навіть якщо така модель добре узгоджується з наявними даними, це не робить її гарною моделлю. Це може означати лише те, що у моделі є достатньо свободи, що вона може описувати практично будь-який набір даних заданого розміру, не отримуючи ніякої справжньої інформації про основне явище. Коли це станеться, модель буде добре працювати для існуючих даних, але не зможе узагальнити ситуацію. Справжньою перевіркою моделі є її здатність робити прогнози в ситуаціях, які раніше незустрічалися.

Перенавчання є основною проблемою в нейронних мережах. Це особливо актуально в сучасних мережах, які часто мають дуже велику кількість ваг і упереджень. Щоб навчитись ефективно, нам потрібен спосіб виявлення, коли відбувається перенавчання. І ми хотіли б мати методи для

зменшення наслідків надмірного використання. Очевидним способом виявлення перенавчання є використання вищезазначеного підходу, відстеження точності даних тесту, як ми тренуємо мережу. Якщо ми бачимо, що точність даних тесту більше не поліпшується, то ми повинні припинити навчання. Зрозуміло, що, строго кажучи, це не обов'язково ознака перенавчання. Можливо, точність даних про тестування та дані тренувань одночасно зупиняються. Тим не менш, прийняття цієї стратегії дозволить запобігти перенавчанню. Дотепер ми використовуємо тренувальні дані (ТД) та тестові дані, ігноруючи дані для перевірки (ВД). Дані для перевірки містить зазвичай містять 10%, зображення яких відрізняються від. Замість того, щоб використовувати тестові дані, щоб запобігти перенавчанню, ми будемо використовувати валідаційні дані. Ми будемо вирахувати точність класифікації на валідаційних даних в кінці кожної епохи. Коли точність класифікації насичена, ми припиняємо навчання. Ця стратегія називається ранньою зупинкою. Звичайно, на практиці ми не відразу знаємо, коли точність насичується. Навпаки, ми продовжуємо навчання, поки не впевнені, що точність насичено. Звичайно, це ні в якому разі не відповідає на питання, чому ми використовуємо ВД, щоб запобігти перевизначенню, а не тестові дані. Щоб зрозуміти, чому, вважають, що при встановленні гіперпараметрів ми, ймовірно, спробуємо різні варіанти гіперпараметрів. Якщо ми встановимо гіпер-параметри на основі оцінок тестових даних, можливо, ми завершимо накладання наших гіпер-параметрів. Тобто, ми можемо в кінцевому підсумку знайти гіперпараметри, які відповідають особливим особливостям тестових даних, але де продуктивність мережі не буде узагальнена на інші набори даних. Ми уникаємо цього, визначивши гіперпараметри за допомогою ВД. Тоді, коли ми отримаємо гіперпараметри, які ми хочемо, ми робимо остаточну оцінку точності, використовуючи тестові дані. Це дає нам упевненість у тому, що наші результати на є справжньою мірою того, наскільки добре наша нейронна мережа узагальнює. Іншими словами, ви можете думати про дані перевірки як типу навчальних даних, які допомагають нам вивчати

хороші гіперпараметри. Цей підхід до пошуку хороших гіперпараметрів іноді називають методом витримки.

Збільшення обсягу навчальних даних - це один із способів скорочення перенавчання. Існують інші способиподолати перенавчання. Один з можливих підходів - зменшити розмір мережі. Проте, великі мережі мають потенціал бути більш потужними, і тому це варіант, яким ми б лише знехтували.

На щастя, існують інші методи, які дозволяють зменшити надмірне використання, навіть якщо у нас є фіксована мережа та фіксовані дані для тренувань. Вони відомі як методи регуляризації. Один з найпоширеніших методів регуляризації, яка іноді називається "розкидання ваги" або регуляризація L2. Ідея регуляризації L2 полягає в тому, щоб додати додатковий елемент до функції витрат, елемент, що називається елементом регуляризації. Ось регуляризована крос-ентропія:

$$C = -\frac{1}{n}\sum_{xj}\left[y_j \ln a_j^L + (1-y_j)\ln(1-a_j^L)\right] + \frac{\lambda}{2n}\sum_w w^2.$$

Перший елемент - це звичайний вираз для крос-ентропії. Але ми додали другий елемент, а саме суму квадратів всіх ваг у мережі. Це масштабується коефіцієнтом $\lambda / 2n$, де $\lambda > 0$ відомий як параметр регуляризації, а n, розмір нашого тренувального набору. Варто також відзначити, що термін регуляризації не включає упередження.

Інтуїтивно, ефект регуляризації полягає в тому, щоб зробити так, щоб мережа віддавала перевагу навчанню малих ваг. Великі ваги навчаються лише у тому випадку, якщо вони значно покращують першу частину функції витрат. Іншими словами, регулялізацію можна розглядати як спосіб компенсації між знаходженням малих ваг і мінімізацією первісної функції витрат. Відносна важливість двох елементів компромісу залежить від значення λ: коли λ мала, ми вважаємо за краще мінімізувати початкову функцію вартості, але коли λ є великою, ми віддаємо перевагу малим вагам.

Насправді не зовсім зрозуміло, чому створення такого компромісу має допомогти зменшити надмірне використання! Але виявляється, що це так. Ми розглянемо питання про те, чому це допомагає у наступному розділі.

Спочатку потрібно з'ясувати, як застосовувати наш алгоритм навчання стохастичного спуску градієнта в регуляризованій нейронній мережі. Зокрема, ми повинні знати, як обчислити часткові похідні ∂C / ∂w і ∂C / ∂b для всіх ваг і зміщення в мережі.

$$\frac{\partial C}{\partial w} = \frac{\partial C_0}{\partial w} + \frac{\lambda}{n} w$$
$$\frac{\partial C}{\partial b} = \frac{\partial C_0}{\partial b}.$$

Умова ∂C0 / ∂w та ∂C0 / ∂b можна обчислити, використовуючи зворотне поширення. І тому ми бачимо, що легко розрахувати градієнт регуляризованої функції витрат: просто використавши зворотнє розповсюдження, як звичайно, а потім додати λnw до часткової похідної всіх умов ваги. Часткові похідні по відношенню до зсувів не змінюються, і тому правило градієнтного спуску для вивчення зсувів не змінюється.

Описано регуляризацію як спосіб зменшення надмірності та збільшення точності класифікації. Емпірично, під час виконання декількох пробігів мережі, але з різними (випадковими) ініціаціями ваги, виявлено, що нерегулярні спроби іноді будуть "застрягати", очевидно, зачеплені в місцевих мінімумах функції витрат. Результат полягає в тому, що різні запуски іноді дають зовсім різні результати. На відміну від цього, регуляризовані спроби дають набагато більш легко повторювані результати.

Чому це відбувається? Звісно, якщо функція вартості не регламентована, то довжина вагового вектора, імовірно, буде зростати, причому всі інші речі будуть однаковими. З часом це може призвести до дуже великого вагового вектора. Це може призвести до того, що ваговий вектор застрягне, вказуючи його більш-менш в тому ж напрямку, оскільки зміни внаслідок градієнтного спуску лише роблять крихітні зміни у

напрямку, коли довжина довга. Це явище ускладнює алгоритм навчання для належного вивчення вагового простору і, отже, важче знайти хороші мінімуми функції витрат.

Ми емпірично бачили, що регуляризація допомагає зменшити надмірне навчання. Здається що, малі ваги в певному сенсі мають меншу складність, і тому вони дають прості та потужніші пояснення даних, і тому вони повинні бути кращими. Проте це досить коротка історія, і містить декілька елементів, які, можливо, здаються сумнівними. Щоб це довести, візьмемо простий набір даних, для побудови моделі:

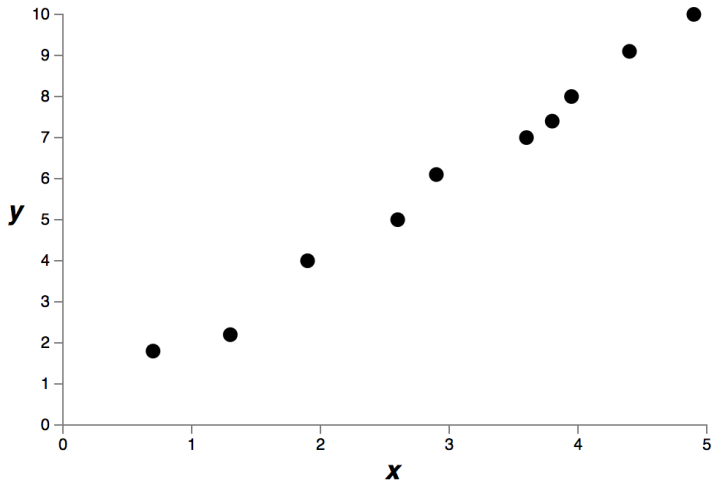

Рисунок 12 - Приклад набору даних

Нашою метою є побудова моделі, яка дозволяє прогнозувати y як функцію x. Ми можемо спробувати використовувати нейронну мережу для побудови такої моделі, але я збираюся зробити щось ще простіше: спробую побудувати модель y як полінома в x. Тепер на графіку вище знаходиться десять балів, що означає, що ми можемо знайти унікальний многочлен 9-го порядку $y = a_0x^9 + a_1x^8 + ... + a^9$, який точно відповідає даним. Ось графік цього полінома (рис. 13).

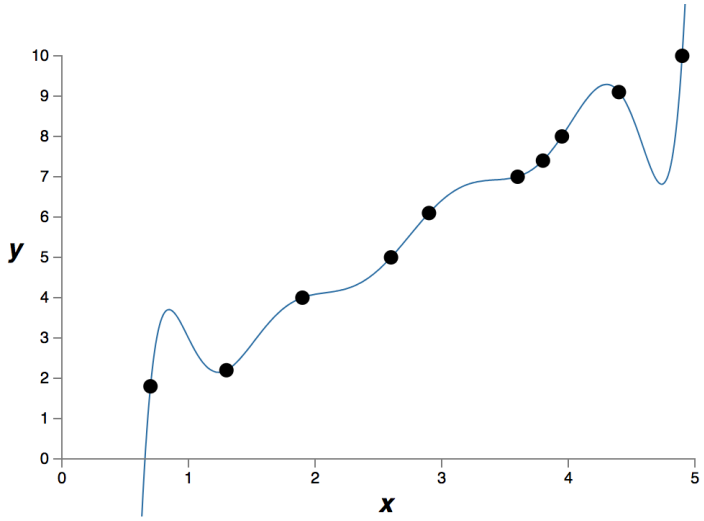

Рисунок 13 - Графік полінома

Це забезпечує точну результат. Але ми також можемо отримати гарну результат за допомогою лінійної моделі y = 2x:

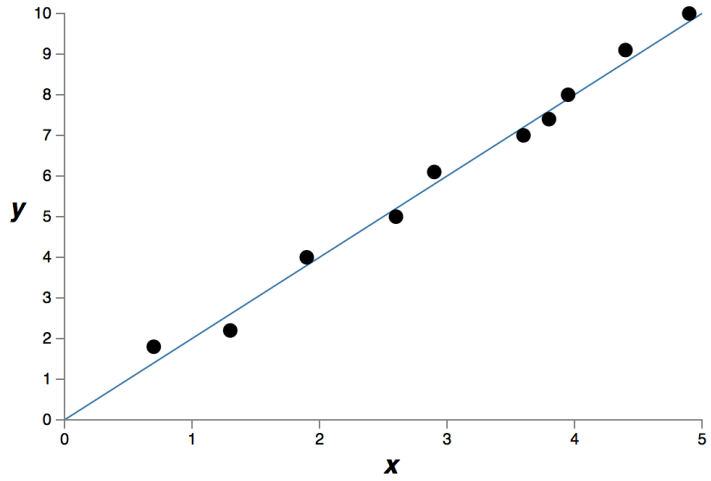

Рисунок 14 - Лінійний розподіл

Яка з них краща модель? Що, більше за все, буде правдою? І яка модель, більше за все, узагальнює інші приклади одного і того ж базового феномена реального світу? Це складні питання. Фактично, ми не можемо точно визначити відповідь на будь-який з наведених вище питань, не маючи набагато більше інформації про основне явище реального світу. Але розглянемо дві можливості: (1) поліноми 9-го порядку - це, по суті, модель, яка дійсно описує феномен реального світу, і модель буде точно узагальнювати; (2) правильна модель є y = 2x.

За наданими даними лише дві невеликі відмінності між двома моделями. Але припустімо, що ми хочемо передбачити значення y, яке відповідає деякому великому значенню x, набагато більшого, ніж будь-яке, показане на графіку вище. Якщо ми намагатимемось зробити це, буде різка різниця між прогнозами двох моделей, оскільки модель 9-го порядку поліноми домінує терміном x^9, а лінійна модель залишається.

Однією точкою зору є те, що в науці ми повинні йти з більш простим поясненням, якщо не змусити цього не робити. Зрештою, малоймовірно, що просте пояснення має відбуватися лише збігом. Швидше за все, ми підозрюємо, що модель повинна виражати деяку основну правду про це явище. У випадку під рукою модель y = 2x + шум здається набагато простішою, ніж $y = a_0x^9 + a_1x^8 + ...$ Було б дивним, якби ця простота відбулася випадково, і тому ми підозрюємо, що y = 2x + шум виражає якусь істинну основу. З цієї точки зору, модель 9-го порядку дійсно просто вивчає наслідки локального шуму. І тому, що модель 9-го порядку ідеально працює для цих конкретних даних, модель не зможе узагальнити інші дані, а шумна лінійна модель буде мати більшу прогностичну силу.

Подивимося, що означає ця точка зору для нейронних мереж. Припустимо, мережа в основному має невеликі ваги, як це буде, як правило, відбувається в регуляризованій мережі. Мала кількість ваг означає, що поведінка мережі не зміниться занадто сильно, якщо ми змінюємо кілька випадкових ваг тут і там. Це ускладнює регуляризовану мережу для вивчення впливу локального шуму на дані. Натомість регуляризована мережа навчається реагувати на типи доказів, які часто зустрічаються у навчальній серії. На відміну від цього, мережа з великими вагами може трохи змінювати свою поведінку у відповідь на невеликі зміни входу. Таким чином, незареєстрована мережа може використовувати великі ваги для вивчення складної моделі, яка містить велику кількість інформації про шум у навчальних даних. У двох словах, регуляризовані мережі змушені будувати відносно прості моделі на основі моделей, які часто зустрічаються в навчальних даних, і є стійкими до вивчення особливостей шуму в

навчальних даних. Надія полягає в тому, що це змусить наші мережі реально вивчати явища, і краще узагальнювати їх від того, що вони вивчають.

2.2.6. ЗМЕНШЕННЯ НАДЛИШКОВОГО НАВЧАННЯ ЗА ДОПОМОГОЮ DROPOUT

Dropout - це принципово інша методика регуляризації. На відміну від регуляризації L1 та L2, dropout не залежить від модифікації функції витрат. На відміну від цього ми змінюємо саму мережу. Опишемо основні механізми роботи відсіву (dropout), перш ніж з'ясувати, чому це працює, і які результати.

Візьмемо повнозв’язну нейронну мережу (рис. 15).

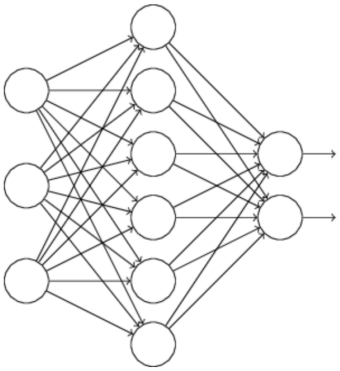

Рисунок 15 - Повнозв’язна нейронна мережа

Припустимо, що ми маємо навчальний вхід x та відповідний бажаний вихід y. Зазвичай ми тренуємося шляхом прямого поширення x через мережу, а потім використовували зворотнє поширення, щоб визначити внесок у градієнт. З відсівом цей процес модифікується. Ми починаємо з випадкового видалення половини прихованих нейронів у мережі, залишаючи вхідні та вихідні нейрони недоторканими. Відсіяні нейрони, тобто тимчасово видалені нейрони, як і раніше присутні у мережі:

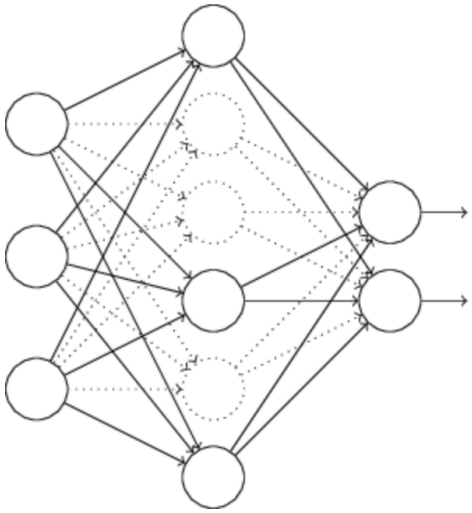

Рисунок 16 - Повнозв’язна нейронна мережа після застосування Dropout

Таким чином ми поширюємо дані через модифіковану мережу, а потім виконуємо зворотне поширення результату, також через модифіковану мережу. Після того, як зробимо це за допомогою міні-партії прикладів, ми оновлюємо відповідні ваги та упередження. Потім ми повторюємо процес, спочатку відновлюючи відсіяні нейрони, потім вибираємо нову випадкову підмножину прихованих нейронів, які потрібно видалити, оцінюючи градієнт для іншої міні-партії та оновлюючи ваги та упередження в мережі. Повторюючи цей процес знову і знову, наша мережа вивчить набір ваг і упереджень. Звичайно, ці ваги та упередження будуть вивчені в умовах, коли половина прихованих нейронів випала. Коли ми фактично запускаємо повну мережу, це означає, що вдвічі більша кількість прихованих нейронів буде активною. Щоб компенсувати це, ми зменшимо вдвічі ваги, що виходять із прихованих нейронів. Ця процедура відсіву може здатися дивною та надмірною. Щоб пояснити, уявімо, що ми готуємо кілька різних нейронних мереж, всі вони використовують однакові навчальні дані. Звичайно, мережі можуть не розпочинатись ідентично, і в результаті після навчання вони іноді можуть дати різні результати. Коли це станеться, ми можемо використати певну схему усереднення або голосування, щоб вирішити, який висновок приймати. Наприклад, якщо ми навчаємо п'ять

мереж, а три з них класифікують зображення до класу "А", то це, напевно, "А". Інші дві мережі, ймовірно, просто роблять помилку. Така схема усереднення часто виявляється потужним (хоча й дорогим) способом зменшення надмірності. Причиною є те, що різні мережі можуть перевитрати по-різному, а усереднення може допомогти усунути такий тип перенавчання. Евристично, коли ми випускаємо різні набори нейронів, це швидше, як ми тренуємо різні нейронні мережі. Отже, процедура відсіву схожа на усереднення ефектів дуже великої кількості різних мереж. Ця методика зменшує складні кооперації нейронів, оскільки нейрон не може покладатися на присутність окремих інших нейронів. Тому він змушений дізнаватися більше надійних функцій, корисних у поєднанні з багатьма різними випадковими підмножинами інших нейронів. Іншими словами, якщо ми думаємо про нашу мережу як про модель, яка робить прогнози, то ми можемо думати про відмову від курсу, як спосіб переконатись, що модель є надійною для втрати будь-якої індивідуальної доказової інформації. У цьому він дещо схоже на регуляризацію L1 і L2, які, як правило, зменшують ваги, і тим самим робить мережу більш надійною, щоб втратити будь-яке індивідуальне з'єднання в мережі. Отже, dropout є особливо корисною функцією для підготовки великих, глибоких мереж, де часто проблема надмірного навчання часто є гострою.

2.2.7. ВИКОРИСТАННЯ ОБРАНИХ МЕТОДІВ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

Для класифікації зображень з соціальної мережі Instagram за гендерною ознакою вирішено використовувати модель MobileNet, яка була тренована на базі даних зображень ImageNet. Такий вибір зумовлений меншою кількістю шарів порівняно з іншими моделями нейронних мереж, що дозволить класифікувати більше даних за менший час. В стандартній конфігурації така модель налаштована на розпізнавання 1000 різних класів зображень, для задачі, яка розглядається в роботі потрібно забезпечити розпізнавання лише двох класів. Зважаючи на це мережа була модифікована

наступним чином: останній шар нейронів розгорнутий в вектор, який підключається до повнозв’язного шару з 256 нейронів до якого застосовано dropout. На виході додано шар з двох нейронів з активаційною функцією softmax, для отримання результату. Для навчання такої мережі потрібно підготувати зображення розміром 256 на 256 пікселів, адже саме такі зображення використовувалися для навчання з бази даних ImageNet. Для навчання використовувався алгоритм стохастичного градієнтного спуску з показником швидкості навчання 0.0001, та інерціним показником 0.9. Шари всередині мережі вже мали треновані ваги, тобто вони мали інформацію про певні абстракції зображень, цей факт був врахований під час навчання. Для навчання використовувалися приблино 16000 зображень чоловіків та жінок різних рас, національностей та віку завантажених за соціальної мережі Instagram та підгодовлених відповідним чином. На першому етапі навчання було треновано тільки частину мережі, яка була додана в кінці остновної мережі, інші ваги не змінювали своїх значень. Після цього треновано перші 20 шарів та продовжено тренування останньої частини мережі. Таке навчання дозволяє уточнити вхідні дані, залишити отримані абстракції високого рівня без змін та налаштувати мережу на класифікацію зображень до двох класів. Використавши вище описані методи, алгоритми та підходи отримано точність класифікації у 85%.

2.3 ВИСНОВКИ

У розділі розглянуто налаштування та використання конкретних методів класифікації зображень та тексту. Для аналізу тексту а саме імен та ніків використана повнозв’язна нейронна мережа з декількох шарів та застосований dropout з ймовірністю видалення нейронів 0.3. Після навчання дана мережа дає точність класифікації у 90%.

Для класифікації зображень використана модель нейронної мережі MobileNet. Розглянуто використання різних алгоритмів навчання, методів оптимізації та покращення, а також способи запобігання надмірного

навчання. Виконана модифікація моделі мережі, підготований набір даних для навчання та обрано ефективний алгоритм навчання: алгоритм стохастичного спуску показником швидкості навчання 0.0001, та інерціним показником 0.9. Поетапне навчання різних частин такої мережі дозволило налаштувати модель на виділення з потрібних зображень абстракцій різного рівня та подальшої їх правильної класифікації. Після навчання така мережа показує точність у 85%.

3. РОЗРОБКА СИСТЕМИ КЛАСИФІКАЦІЇ КОРИСТУВАЧІВ INSTAGRAM ЗА ГЕНДЕРНОЮ ОЗНАКОЮ

Профілі користувачів соціальної мережі Instagram мають багато інформації: повне ім’я, нік, фотографія сторінки, фотографії на сторінці, коментарі, підписи під фотографіями, зв’язки з іншими користувачами. Аналізуючи ці дані можливо класифікувати користувачів до різноманітних класів. Таким чином для класифікації за гендерною ознакою вирішено використовувати повне ім’я, нік, фотографію сторінки та 6 перших фотографій сторінки (з 12 можливих), якщо вони доступні.

Для класифікації зображень та імен використані моделі які були описані раніше у роботі. У розпізнаванні зображень досягнуто точності в 85%, для імен цей показник склав 90%.

Для коректної класифікації повних імен та ніків користувачів, дані які були отримані з Instagram потрібно попередньо обробити. Це зумовлена тим, що в соціальній мережі, яка розглядається дозволено використовувати символи “.”, “_” та цифри в іменах. Таким чином перед класифікацією такі символи були видалені. Поширеним шаблоном ніків є наступні

“dmytro.anastasiev” або “d_m_y_t_r_o”, такі випадки були враховані і після первинної обробки для прикладів згаданих вище отримуємо “dmytro”, що збільшує шанси на успішну класифікацію. Для зображень перед подачею на вхід нейронній мережі змінювався розмір на 256 на 256 пікселів, адже за допомогою зображень такого розміру відбувалося тренування.

Результатом класифікації кожного елемента (імені чи зображення) є два числа від 0 до 1, що в сумі дають 1. Таким чином для отримання кінцевого результату класифікації можна просто визначити суми всіх результатів передбачень для кожного класу, та порівняти їх. Виходячи з того що дані, які розглядаються є різними, тобто одні з них можна класифікувати точніше ніж інші. Зважаючи на це для обчислення кінцевого результату були введені коефіцієнти для кожного вимірювання. Більш точними характеристиками користувача є повне ім’я та фотографія сторінки, тому передбачення по цим даним є точнішим, що дозволяє обрати вищий коефіцієнт для цих вимірювань.

Для демонстрації роботи системи розроблено додаток з веб інтерфейсом, який дозволяє користувачу ввести ім’я користувача соціальної мережі Instagram та отримати результати класифікації за кожною з характеристик (Рис.17.).

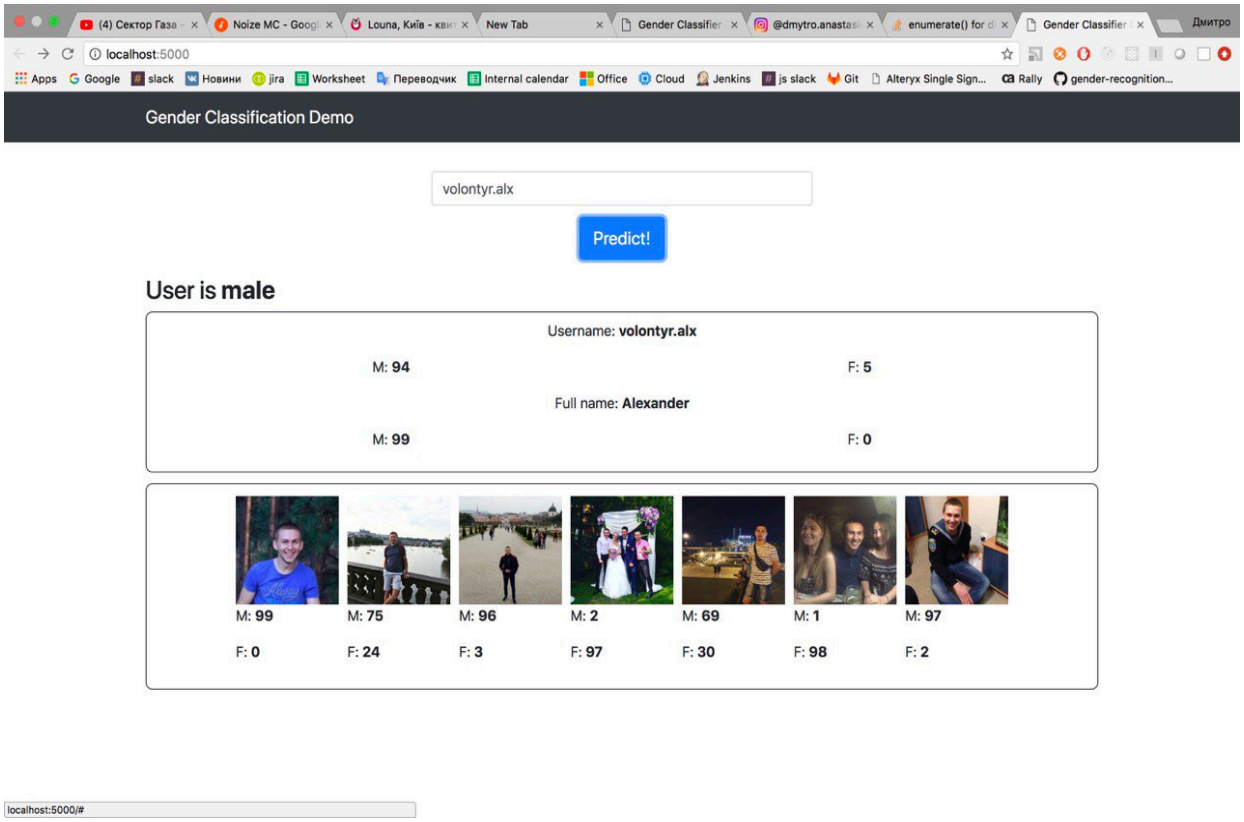

Рисунок 17 - Приклад роботи додатку для класифікації користувачів

На рис. 17. зображено приклад результату класифікації для користувача volontyr.alx. В даному прикладі ім’я було класифіковано вірно, фотографії де зображений один чоловік також вірно. Однією з проблем, з якими стикається система є класифікація фотографій на яких зображено більше однієї людини. Велика ймовірність того, що в цьому випадку фотографію буде класифіковано не вірно, для цього в системі враховується декілька фотографій.

Для оцінки точності роботи системи розроблено тести, за допомогою яких було класифіковано по 700 користувачів соціальної мережі Instagram кожної категорії. За результатами тестування система працює з точністю 87%.

ВИСНОВКИ

У роботі розроблено систему класифікації користувачів соціальної мережі Instagram за гендерною ознакою. Для розробки системи розглянуто та проаналізовано методи та алгоритми класифікації тексту та зображень, в результаті чого вирішено використовувати методи машинного навчання, а саме нейронні мережі. Серед характеристик користувачів для вирішення поставленої задачі використовується повне ім’я користувача, нік, фотографія сторінки та перші доступні 6 фотографій користувача.

Для класифікації зображень вирішено використовувати модель нейронної мережі MobileNet, з внесеними модифікаціями, які дозволяють адаптувати модель для вирішення поставленої задачі. Модель навчено за допомогою попередньо відібраних на підготовлених даних з соціальної мережі Instagram. Після навчання вдалося отримати точність класифікації в 85%.

Для класифікації імен та ніків використано повнозв’язну нейронну мережу навчену на базі даних імен новонароджених дітей США, яка налічує більше ніж 95000 імен. Після навчання вдалося отримати точність в 90%.

Розроблена система поєднує аналіз текстових даних та зображень, що дозволяє точніше визначити стать користувача. Для демонстрації роботи системи розроблено веб інтерфейс, який дозволяє ввести ім’я користувача соціальної мережі та отримати результати класифікації по кожній з досліджених характеристик. Було проведено тестування системи, в ході якого використано по 700 користувачів кожної категорії та в результаті отримано 87% точності класифікації.

У роботі розглянуто успішне передбачення ознак користувачів соціальних мереж, базуючись на наявних ознаках, що дозволяє класифікувати їх за певними групами, при чому це можливо робити не тільки за гендерною ознакою, а також за тими характеристиками, які цікаві для вирішення конкретних задач: вік, національність, уподобання, тощо. Вирішення таких задач класифікації може використовуватися для реалізації розповсюдження рекламної інформації орієнтованої на користувача, а також для пошуку кваліфікованих спеціалістів, які володіють певними вміннями, які потрібні для вирішення поставлених задач.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Raul R. Neural Networks a systematic introduction / Rojas Raul. – Berlin: Springer-Verlag, 1996. – (Neural network) // Режим доступу: http://page.mi.fu-berlin.de/rojas/neural/neuron.pdf
2. Степанов Л. В. Выбор функции активации и обучение нейронной сети / Л. В. Степанов. – Москва: Академия Естествознания, 2009. // Режим доступу: https://www.monographies.ru/ru/book/section?id=2465
3. Michael N. Neural Networks and Deep Learning [Електронний ресурс] / Nielsen Michael. – 2013. – Режим доступу до ресурсу: http://neuralnetworksanddeeplearning.com/.
4. Beyond the Top 1000 Names [Електронний ресурс] – Режим доступу до ресурсу: https://www.ssa.gov/oact/babynames/limits.html.
5. Keras documentation [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/.
6. Scikit-learn documentation [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/
7. Neural networks by Christos Stergiou and Dimitrios Siganos [Електронний ресурс] – Режим доступу до ресурсу: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html
8. Artificial intelegent - Neural networks [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm
9. Introduction neural networks and deep learning [Електронний ресурс] – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2018/10/introduction-neural-networks-deep-learning/